AN APPROACH TO VALIDATION OF DEDUCTIVE CONCEPTUAL MODELS

(preliminary version)

Dolors Costal

Universitat Politècnica de Catalunya
Facultat d'Informàtica
Pau Gargallo 5
08028 Barcelona
Catalunya
Tel. 34-3-4017324
Fax. 34-3-4017040
e-mail: dolors@lsi.upc.es

ABSTRACT

We propose an approach to validation of deductive conceptual models. This validation is performed through plan generation. The objective of plan generation is to determine sequences of events that lead to a certain goal state from a certain initial state. Our approach generates a plan in n stages obtaining at each of them the events of the plan that occur at each time point. At each stage, we use the internal events model, obtained from the deductive conceptual model, which explicitly defines insertions and deletions of information induced by previous occurrence or absence of external events. An extension of SLDNF procedure can then be used to obtain sequences of external events that lead to the goal state.

This work has been partially supported by PRONTIC program project TIC 680

KEYWORDS

Deductive Conceptual Models, Requirements Validation, Plan Formation

july 1991

The paper is organised as follows. Next section defines basic concepts of DCMs and introduces a typical planning example (the blocks world example) that will be used throughout the paper. Section 3 presents the IEM. Section 4 describes our approach. Finally, section 5 gives some conclusions.

2. DEDUCTIVE CONCEPTUAL MODELS

There are several languages for deductive conceptual modelling of information systems. Among them are CIAM [GKB 82], DADES [Oli 82] and IPL [Gri 82]. These languages differ in many aspects, but all share a common approach. In this paper, we will abstract from the details of specific languages and try to characterize the deductive conceptual modelling approach in a first order logic framework.

Time plays a major role in this approach. Every possible information i about the Universe of Discourse (UoD) is associated with a time point T(i), which states when the information holds in the UoD. It can be called the occurrence or observation time. This time point is a component of the information proper, so informations are self-describing in this respect. We will assume that occurrence times are always expressed in a unique time unit (such as second, day, etc.) small enough to avoid ambiguities.

By life span T of an information system we mean the time interval in which the system operates. It is defined as an ordered set of consecutive time points $T = \{T_0,...,T_f\}$, where T_0 and T_f are the initial and final times, respectively, and where each $t \in T$ is expressed in the given time unit. We can then say that, for any information i, $T(i) \in T$.

A DCM consists of a set B of base predicates, a set D of derived predicates, a set DR of deduction rules, a set IC of integrity constraints, and a set OR of output requirements. Each of them except OR is described in the following, and illustrated with references to the DCM example of figure 1. We do not describe, nor give examples of OR because they are not needed for our purposes.

1. INTRODUCTION

This paper deals with the problem of validation of information systems requirements. By validation we mean checking that the specification really reflects user needs and his intended statements about the information system, that is, checking that the specification corresponds to user's requirements.

There are different approaches to the validation problem. One of them is the automatic generation of prototypes. Another approach, that we will study in this paper, is to provide a tool that facilitates reasoning about the information system specification through plan generation. The objective of plan generation is to determine sequences of events that lead to a certain goal state from a certain initial state. This tool will be able to answer questions about the specification, such as: Is it possible to reach a state where the stock of a product is negative?, or in another context, is it possible to reach a state where the number of students enrolled in a course is greater than its capacity?. It will be possible to validate the specification by making the appropriate questions.

There are two basic approaches in the area of conceptual modeling of information systems: operational and deductive. Here we will focus on the validation of Deductive Conceptual Models (DCMs). A detailed comparison of the operational and deductive approaches can be found in [BuO 86] and [Oli 86]. The deductive approach presents many advantages over the operational one but DCMs are much more difficult to implement.

There are works similar to ours but in the operational approach: [VeF 85], [FuM 84], [FuC 90]. In these papers a tool is described that given a state checks whether there exists a sequence of operations leading to that state.

Our problem is strongly related with that of "Plan formation" in the Artificial Intelligence area, [Kow 79], [Esh 88]. We are interested on the aplication of planning techniques to DCMs validation.

In this paper, we present an approach to the validation of DCMs by planning. The approach generates a plan in n stages obtaining at each of them the events of the plan that occur at each time point. At each stage we use the Internal Events Model (IEM) that can be derived from the DCM and that was developed [Oli 89] for the design of information systems from DCMs. The IEM explicitly defines insertions and deletions of information induced by previous occurrence or absence of external events. An extension of SLDNF procedure can then be used to obtain sequences of external events that lead to the goal state. The problem to solve at each stage is similar to that of view updating in deductive databases, [Dec 89], [KaM 90], [TeO 91]. Our approach is an adaptation of the method introduced in [TeO 91].

Base predicates

place(object,time)

manip(object,time)

init(object,on-object,time)

transfer(object,from-object,to-object,time)

Derived predicates

DR.1 $on(x,z,t) \leftarrow go(x,z,t1) \land t1 \leq t \land \neg leave(x,t1,t)$

DR.2 $leave(x,t1,t) \leftarrow go(x,z,t1) \land t1 < t \land go(x,w,t2) \land t2 > t1 \land t2 \leq t$

DR.3 $clear(x,t) \leftarrow object(x,T_0) \land \neg under(x,t)$

DR.4 $under(x,t) \leftarrow object(y,T_0) \land on(y,x,t)$

DR.5 $object(x,t) \leftarrow manip(x,t)$

DR.6 $object(x,t) \leftarrow place(x,t)$

DR.7 $go(x,z,t) \leftarrow init(x,z,t)$

DR.8 $go(x,z,t) \leftarrow transfer(x,y,z,t)$

Integrity constraints

IC.1 $ic1(t) \leftarrow transfer(x,y,z,t) \land \neg on(x,y,t\text{-}1)$

IC.2 $ic2(t) \leftarrow transfer(x,y,z,t) \land \neg clear(x,t\text{-}1)$

IC.3 $ic3(t) \leftarrow transfer(x,y,z,t) \land \neg clear(z,t\text{-}1)$

IC.4 $ic4(t) \leftarrow transfer(x,y,z,t) \land x=z$

IC.5 $ic5(t) \leftarrow transfer(x,y,z,t) \land transfer(u,v,w,t) \land x \neq u$

IC.6 $ic6(t) \leftarrow transfer(x,y,z,t) \land transfer(u,v,w,t) \land y \neq v$

IC.7 $ic7(t) \leftarrow transfer(x,y,z,t) \land transfer(u,v,w,t) \land z \neq w$

IC.8 $ic8(t) \leftarrow transfer(x,y,z,t) \land \neg manip(x,T_0)$

IC.9 $ic9(t) \leftarrow transfer(x,y,z,t) \land \neg object(z,T_0)$

IC.10 $ic10(t) \leftarrow transfer(x,y,z,t) \land t \leq T_0$

IC.11 $ic11(t) \leftarrow init(x,y,t) \land init(u,v,t) \land x \neq u \land y=v$

IC.12 $ic12(t) \leftarrow init(x,y,t) \land init(u,v,t) \land x=u \land y \neq v$

IC.13 $ic13(t) \leftarrow init(x,y,t) \land \neg manip(x,T_0)$

IC.14 $ic14(t) \leftarrow init(x,y,t) \land \neg object(y,T_0)$

IC.15 $ic15(t) \leftarrow init(x,y,t) \land t \neq T_0$

IC.16 $ic16(t) \leftarrow manip(x,t) \land t \neq T_0$

IC.17 $ic17(t) \leftarrow place(x,t) \land t \neq T_0$

IC.18 $ic(t) \leftarrow ic1(t)$

...

IC.34 $ic(t) \leftarrow ic17(t)$

Figure 1. Example of Deductive Conceptual Model

2.1 Base predicates

Base predicates model the external event types of the UoD. They are the inputs to the information system. Each fact of a base predicate, called base fact, corresponds to an occurrence of an external event. We assume, by convention, that the last term of a base fact gives the time when the external event occurred. If $p(a_1,...,a_n,t_i)$ is a fact we say that $p(a_1,...,a_n)$ is true or holds at t_i.

In the example of figure 1 we have four base predicates: place, manip, init and transfer. Place, manip and init can only happen at the initial time (this will be enforced by IC.17, IC.16 and IC.15, respectively). A base fact place(o1,T_0) means that at the initial time, o1 is a place. A base fact manip(o1,T_0) reports that at the initial time, o1 is manipulatable. A base fact init(o1,o2,T_0) means that at the initial time, object o1 is on object o2. A base fact transfer(o1,o2,o3,t1) indicates that object o1 that was on object o2 has been transferred from o2 to o3 at time t1.

2.2 Derived predicates

Derived predicates model the relevant types of knowledge about the UoD. Each fact of a derived predicate, called derived fact, represents an information about the state of the UoD, at a particular time point. We will also assume that the last term of a derived fact gives the time when the information holds. The semantics of a derived predicate is given by its deduction rules.

In the example there are six derived predicates: on, leave, clear, under, object and go. For instance: on(o1,o2,t1) means that object o1 is on object o2 at time t1 and clear(o1,t1) indicates that object o1 has no object on it at time t1.

2.3 Integrity constraints

Integrity constraints are closed formulae that base and/or derived facts must satisfy to be consistent. An integrity constraint can only be falsified by the presence (or absence) of new base facts, and it is assumed that some mechanism of integrity constraints enforcement will reject (or require) those facts to maintain the informations consistent.

We deal here with constraints that have the form of a denial $\leftarrow L_1 \wedge \ldots \wedge L_n$ where the L_i are literals, and variables are assumed to be quantified over the whole formula. For the sake of uniformity we associate to each integrity constraint an inconsistency predicate "icn". The above denial would be rewritten as: $ic1 \leftarrow L_1 \wedge \ldots \wedge L_n$. The semantics of an integrity constraint is given by its deduction rules.

In the example, there are seventeen integrity constraints. For instance: ic1(t) means that if there is a transfer of object x from object y to object z at time t then x should be on y at time t-1, ic2(t) and ic3(t) mean that if there is a transfer of object x from object y to object z then x and z should be clear at t-1, respectively. An additional inconsistency predicate ic is defined in order to simplify the reference to the whole set of inconsistency predicates. Thus, ic(t) holds at time t if some icj(t), j=1...17, holds at t.

2.4 Deduction rules

There are one or more deduction rules for each derived or inconsistency predicate. Let $p(x_1,\ldots,x_n,t)$ be a derived or inconsistency predicate, with n+1 terms. A deduction rule for p has the form $p(x_1,\ldots,x_n,t) \leftarrow \phi$ where ϕ is a literal or a conjunction of (positive or negative) literals, and all variables are assumed to be universally quantified over the whole formula.

The terms in the rule head must be distinct variables. The terms in ϕ (rule body) must be constants or variables. We assume every rule to be range-restricted, i.e. every variable occurring in the head, or in a negative literal in ϕ, occurs in a positive literal in ϕ as well. We also assume every rule to be time-restricted. This means that, for every base or derived predicate q occurring in the body as a positive literal q(...,t1), the condition $\phi \rightarrow t1 \leq t$ must hold. This condition ensures that $p(x_1,\ldots,x_n,t)$ is defined in terms of q-facts holding at time t or before.

3. THE INTERNAL EVENTS MODEL

In [Oli 89] and [San 90] it is presented a formal method to derive from a DCM a new model called the Internal Events Model (IEM). In this section we will shortly review the concepts and terminology of internal events predicates and internal events rules which form the IEM. In next section, IEM will be used for generating plans.

3.1 Internal events predicates

There are two classes of internal events: insertion and deletion. To each base, derived or inconsistency predicate p corresponds an insertion internal event predicate ιp, with the same number of terms, and to some of the derived predicates q corresponds a deletion internal event δq, with the same number of terms.

If p is a base predicate then ιp(...,t) fact represents the occurrence of an external event at time t. There are no deletion internal events for base predicates.

If p is a derived predicate then ιp(...,t) and δp(...,t) facts represent insertions and deletions of p at time t induced by the occurrence or absence of external events at times $\leq$ t. A derived predicate may be defined in such a way that when it has become true, it remains true forever. For this derived predicate, corresponding deletion internal event is not defined.

If p is an inconsistency predicate then ιp(...,t) fact represents violation of an integrity constraint. For inconsistency predicates δp is nonsense and thus it is not defined.

We say that an internal event predicate, ιp or δp, is base, derived or inconsistency if p is base, derived or inconsistency, respectively.

3.2 Internal events rules

There are internal events rules defined for each derived or inconsistency internal event predicate. In [Oli 89] a formal method is shown to derive internal events rules. Internal events rules allow us to deduce at any time which internal events occur at that time.

The body of internal events rules obtained is a conjunction of (positive or negative) literals. Each literal can be a current literal corresponding to an internal event predicate, a past literal corresponding to a base, derived or inconsistency predicate or a literal corresponding to an evaluable predicate. We understand by past literals those whose time variables range over a set not including t and by current literals those whose time variables range only over {t}.

Figure 2 shows the IEM that corresponds to the DCM of our example.

IDR.1 $\iota on(x,z,t) \leftarrow \iota go(x,z,t)$

IDR.2 $\delta on(x,z,t) \leftarrow \iota transfer(x,z,w,t)$

IDR.3 $\iota leave(x,t1,t) \leftarrow \iota transfer(x,z,w,t)$

IDR.4 $\iota clear(x,t) \leftarrow \delta on(y,x,t)$

IDR.5 $\delta clear(x,t) \leftarrow \iota on(y,x,t)$

IDR.6 $\iota under(x,t) \leftarrow \iota on(y,x,t)$

IDR.7 $\delta under(x,t) \leftarrow \delta on(y,x,t)$

IDR.8 $\iota object(x,t) \leftarrow \iota manip(x,t)$

IDR.9 $\iota object(x,t) \leftarrow \iota place(x,t)$

IDR.10 $\iota go(x,z,t) \leftarrow \iota init(x,z,t)$

IDR.11 $\iota go(x,z,t) \leftarrow \iota transfer(x,y,z,t)$

IDR.12 $\iota ic1(t) \leftarrow \iota transfer(x,y,z,t) \wedge \neg on(x,y,t-1)$

IDR.13 $\iota ic2(t) \leftarrow \iota transfer(x,y,z,t) \wedge \neg clear(x,t-1)$

IDR.14 $\iota ic3(t) \leftarrow \iota transfer(x,y,z,t) \wedge \neg clear(z,t-1)$

IDR.15 $\iota ic4(t) \leftarrow \iota transfer(x,y,z,t) \wedge x=z$

IDR.16 $\iota ic5(t) \leftarrow \iota transfer(x,y,z,t) \wedge \iota transfer(u,v,w,t) \wedge x\neq u$

IDR.17 $\iota ic6(t) \leftarrow \iota transfer(x,y,z,t) \wedge \iota transfer(u,v,w,t) \wedge y\neq v$

IDR.18 $\iota ic7(t) \leftarrow \iota transfer(x,y,z,t) \wedge \iota transfer(u,v,w,t) \wedge z\neq w$

IDR.19 $\iota ic8(t) \leftarrow \iota transfer(x,y,z,t) \wedge \neg manip(x,T_0)$

IDR.20 $\iota ic9(t) \leftarrow \iota transfer(x,y,z,t) \wedge \neg object(z,T_0)$

IDR.21 $\iota ic10(t) \leftarrow \iota transfer(x,y,z,t) \wedge t\leq T_0$

IDR.22 $\iota ic11(t) \leftarrow \iota init(x,y,t) \wedge \iota init(u,v,t) \wedge x\neq u \wedge y=v$

IDR.23 $\iota ic12(t) \leftarrow \iota init(x,y,t) \wedge \iota init(u,v,t) \wedge x=u \wedge y\neq v$

IDR.24 $\iota ic13(t) \leftarrow \iota init(x,y,t) \wedge \neg manip(x,T_0)$

IDR.25 $\iota ic14(t) \leftarrow \iota init(x,y,t) \wedge \neg object(y,T_0)$

IDR.26 $\iota ic15(t) \leftarrow \iota init(x,y,t) \wedge t\neq T_0$

IDR.27 $\iota ic16(t) \leftarrow \iota manip(x,t) \wedge t\neq T_0$

IDR.28 $\iota ic17(t) \leftarrow \iota place(x,t) \wedge t\neq T_0$

IDR.29 $\iota ic(t) \leftarrow \iota ic1(t)$

...

IDR.45 $\iota ic(t) \leftarrow \iota ic17(t)$

Figure 2. IEM of the example

4. OUR APPROACH

Returning to our problem, we want to generate a plan that leads to a goal state, that we want the information system to reach, from a certain initial state.

The goal state should be characterized as a set of informations that must or must not hold in that goal state. In a DCM this is defined as a conjunction of literals corresponding to base, derived, inconsistency or evaluable predicates, that can be negated or not. Terms in those literals are constants or variables. We assume the goal to be range-restricted, i.e. every variable occurring in a negative literal, occurs in a positive literal as well. A time point, tg, is associated to any goal state, such that, tg is the minimum time that for every positive literal, p(...,t1), in the goal state, $t1 \leq tg$ holds. For example:

$$on(A,B,t) \wedge on(B,F,t)$$

where on is a derived predicate and tg=t.

It is important to remark that if the generated plan has to satisfy the integrity constraints, the goal state should include that none of the inconsistency predicates holds. This will ensure that the plan generated satisfies the integrity constraints. From this, the example above should be reformulated in order to obtain a plan without integrity constraint violation:

$$on(A,B,t) \wedge on(B,F,t) \wedge \neg\, ic(t)$$

We will use this example, represented in figure 3, throughout the paper.

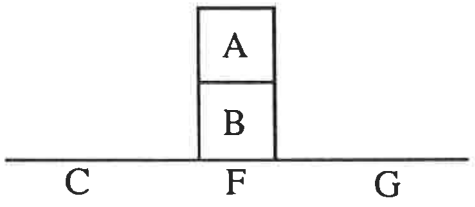


Figure 3. Goal state

The initial state is defined as a set of literals corresponding to base predicates. Terms in those literals must be constants. A time point, ti, is associated to any initial state, such that, ti is the minimum time that for every literal, p(...,t1), in the initial state, $t1 \leq ti$ holds. The initial state may be empty, in that case, the associated time point is T_0, in which the life of the system begins. The initial state of our example, represented in figure 4, is:

$\{place(C,T_0), place(F,T_0), place(G,T_0), manip(A,T_0), manip(B,T_0), init(B,C,T_0), init(A,B,T_0)\}$

In our case, the associated time point is T_0.

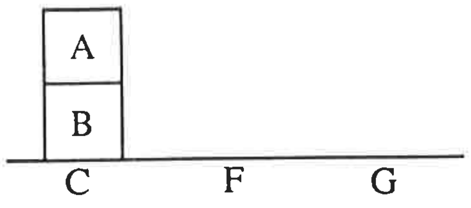


Figure 4. Initial state

A plan is a set of external events of the UoD that leads to the goal state at time tg from the initial state at time ti. In a DCM these external events are modeled by base predicates so the plan to generate will be a set of literals corresponding to base internal events predicates. The plan we should obtain in our example is:

$$P= \{\iota transfer(A,B,G,T_0+1),\ \iota transfer(B,C,F,T_0+2),\ \iota transfer(A,G,B,T_0+3)\}$$

Once defined the departure and arrival elements of our problem, we are going to present the strategy chosen to solve it.

A plan evolves through some consecutive time points, beginning at ti+1 and ending at the time of the final state, tg. We will look at it as n consecutive smaller plans ($T_{ti+1},\ldots,T_{tg-2},T_{tg-1},T_{tg}$) that consider only the transition from an state at time t to an state at the following time. Each T_j contains base predicates that must happen at time j. We illustrate this in figure 5.

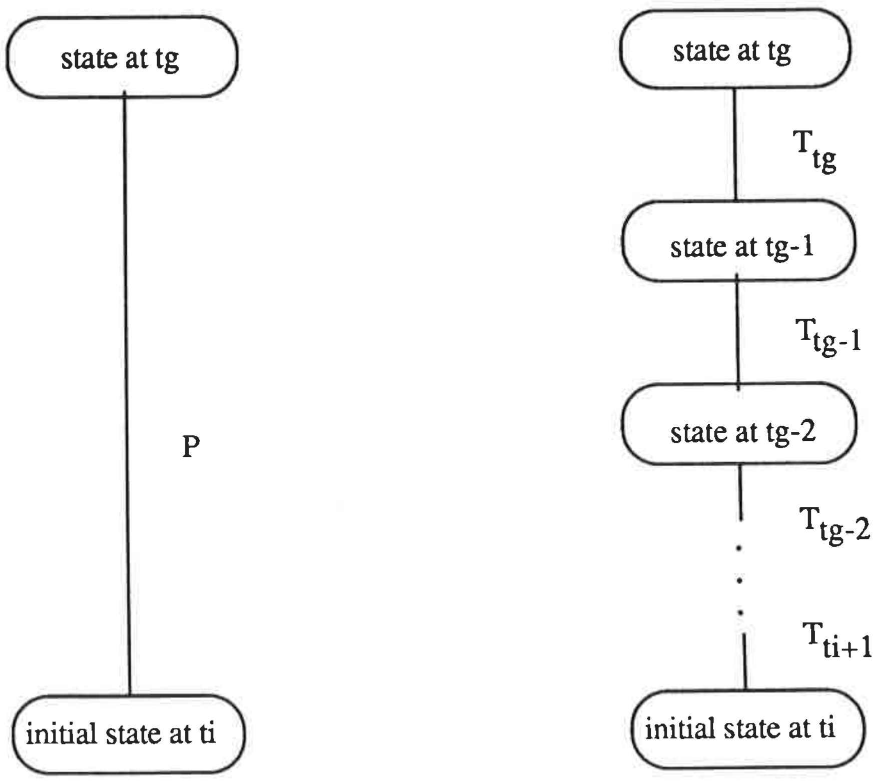


Figure 5

From this point of view we can reduce the problem of generating a general plan to the n smaller problems of given a goal state, G, at a current time to find the previous goal state, PG, at the previous time and the piece of plan, T, that makes the transition from that previous goal state to the current goal state.

In other words, our problem will be solved in n stages. At each of them we depart from a current goal state G and we obtain a previous goal state PG and a transition T. G is a conjunction of literals corresponding to base, derived, inconsistency or evaluable predicates that can be negated or not, and whose time variables range over a set including t, being t the time at the current state. PG is a set of literals corresponding to base, derived, inconsistency or evaluable predicates that can be negated or not, and whose time variables range over a set not including t (past), being t the time at the current state. T is a set of literals, corresponding to base internal event predicates whose time variables range only over current time {t}, that makes the transition from one state to the other. This will be done n times until the previous goal reached, PG, holds at the initial state.

Note that we will generate a plan in the inverse order it should be executed. For example, for plan P={ιtransfer(A,B,G,T_0+1),ιtransfer(B,C,F,T_0+2),ιtransfer(A,G,B,T_0+3)}, we will obtain at first stage: T_{tg}={ιtransfer(A,G,B,T_0+3)}, at second stage: T_{tg-1}={ιtransfer(B,C,F,T_0+2)} and at third stage: T_{tg-2}={ιtransfer(A,B,G,T_0+1)}

The problem we have to solve at each stage has strong similarities with that of view updating in deductive databases. (We can identify G to the update request, T to the transaction obtained to perform the update and PG to the deductive database state before the update). Solutions discovered in that field can be of help if adapted to the context of DCMs [Dec 89], [KaM 90], [TeO 91]. Our approach is an adaptation of the events method for view updating in deductive databases introduced in [TeO 91].

At each stage we will do the following steps:
1) Check satisfaction of the current goal state G at the initial state.
2) Obtain transition rules of G.
3) Generate T and PG.
4) Ensure the consistency of T.
5=1) Check satisfaction of PG at the initial state.

These steps are described in the following subsections.

4.1 Check satisfaction of the current goal state G at the initial state

Current goal state G and initial state are already determined. From this and from deduction rules of the DCM, we can deduce by SLDNF resolution whether current goal state holds at the initial state. If G holds at the initial state, it means that we have completed the plan, otherwise we continue with following steps.

At first stage, terms in G may be variables. It is possible that G holds at the initial state for certain values of those variables.

In our example, at first stage, we should check the satisfaction of the following goal: $on(A,B,t) \wedge on(B,F,t) \wedge \neg ic(t)$, which does not hold at the initial state.

4.2 Obtain transition rules of G

Transition rules of a rule $P \leftarrow L_1 \wedge ... \wedge L_n$ is a set of equivalent rules obtained replacing each literal L_i (i=1...n) corresponding to a base, derived or inconsistency predicate and whose time variable ranges over a set including t by an equivalent expression containing internal events predicates whose time variables range only over {t} (current) and base, derived or inconsistency predicates whose time variables range over a set not including t (past). This transformation was introduced in [Oli 89] where a full description of it is given.

If we associate a goal predicate GP to the current goal state G, we can obtain transition rules of this predicate. In general, we obtain m transition rules corresponding to predicates $GP_1, ..., GP_m$, such that, $G \leftarrow GP_i$, i=1...m. Each GP_i expresses G in terms of past base, derived or inconsistency predicates and current internal event predicates. GP_i of the example are shown below:

$GP \leftarrow on(A,B,t) \wedge on(B,F,t) \wedge \neg ic(t)$

$GP_1 \leftarrow \iota on(A,B,t) \wedge \iota on(B,F,t) \wedge \neg ic(t\text{-}1) \wedge \neg \iota ic(t)$
$GP_2 \leftarrow \iota on(A,B,t) \wedge on(B,F,t\text{-}1) \wedge \neg \delta on(B,F,t) \wedge \neg ic(t\text{-}1) \wedge \neg \iota ic(t)$
$GP_3 \leftarrow on(A,B,t\text{-}1) \wedge \neg \delta on(A,B,t) \wedge \iota on(B,F,t) \wedge \neg ic(t\text{-}1) \wedge \neg \iota ic(t)$
$GP_4 \leftarrow on(A,B,t\text{-}1) \wedge \neg \delta on(A,B,t) \wedge on(B,F,t\text{-}1) \wedge \neg \delta on(B,F,t) \wedge \neg ic(t\text{-}1) \wedge \neg \iota ic(t)$

In GP_i past and current information appears in different literals. In the rules of the IEM, that will be used in following steps, past and current information also appears separately. Due to this, at each stage we will be able to concentrate on obtaining the piece of the plan corresponding to the current time without interferences from goals that must be satisfied at previous times.

Each GP_i is a different transformation of the current goal state G. For example, GP_1 states that we could reach GP (and hence G) by inserting A on B at t ($\iota on(A,B,t)$), B on F at t ($\iota on(B,F,t)$), provided that the previous state is consistent ($\neg ic(t-1)$) and no violations of the integrity constraints happen in the transition ($\neg \iota ic(t)$). Each GP_i may lead to different plans. In our example, we will show the case of selecting GP_2.

4.3 Generate T and PG

T and PG are sets of literals such that using SLDNF resolution, the goal $\{\leftarrow GP_i\}$ succeeds from input set $IEM \cup T \cup PG$.

The transition T and the previous goal PG can be obtained by having some failed derivation of $IEM \cup \{\leftarrow GP_i\}$ succeed. This is effected by including in T a ground instance of each current and positive literal corresponding to a base internal event predicate selected during the derivation and by including in PG each past literal corresponding to a base, derived or inconsistency predicate or literal corresponding to an evaluable predicate, selected during the derivation.

A condition set C will also be obtained during the derivation process. It will contain the denials $\leftarrow D$ such that $\neg D$ is a current and negative literal corresponding to an internal event predicate selected during the derivation. C is only necessary to know which conditions T must satisfy. In the step described in section 4.4 the consistency of T and C is ensured by additions to T, PG and C when necessary.

We will call this derivation "constructive derivation" and a formal description of it is given in this section.

Skolemisation

As mentioned before, depending on the literal selected during the derivation, a ground instance of that literal may have to be added to T or PG.

If the selected literal is not ground we would have to add to T or PG an existentially quantified literal that we cannot use directly in a resolution based system. For example, a selected literal $\iota P(x,y,t)$ means adding $\exists x,y,t\ \iota P(x,y,t)$.

Thus we have to skolemise the literal [Esh 88], that is, replace all existentially quantified variables in it with skolem constants (in the rest of the paper symbols as X^s, Y^s, T^s, ... will be skolem constants and s(L) will be the result of skolemising literal L). For example: $s(\iota p(x,y,t))=\iota p(X^s,Y^s,T^s)$. Note that if the selected literal L is ground then $s(L)=L$.

Example

In the previous section, we selected GP_2 as the transformation of G, that we would use to generate T and PG. We continue the example with the constructive derivation from GP_2:

$$\begin{array}{l}
\leftarrow \underline{\iota on(A,B,t)} \wedge on(B,F,t\text{-}1) \wedge \neg\delta on(B,F,t) \wedge \neg ic(t\text{-}1) \wedge \neg\iota ic(t) \\
\qquad\qquad 1 \quad | \quad (IDR.1) \\
\leftarrow \underline{\iota go(A,B,t)} \wedge on(B,F,t\text{-}1) \wedge \neg\delta on(B,F,t) \wedge \neg ic(t\text{-}1) \wedge \neg\iota ic(t) \\
\qquad\qquad 2 \quad | \quad (IDR.11) \\
\leftarrow \underline{\iota transfer(A,y,B,t)} \wedge on(B,F,t\text{-}1) \wedge \neg\delta on(B,F,t) \wedge \neg ic(t\text{-}1) \wedge \neg\iota ic(t) \\
\qquad\qquad 3 \quad | \quad T=\{\iota transfer(A,Y^s,B,T^s)\} \\
\leftarrow \underline{on(B,F,T^s\text{-}1)} \wedge \neg\delta on(B,F,T^s) \wedge \neg ic(T^s\text{-}1) \wedge \neg\iota ic(T^s) \\
\qquad\qquad 4 \quad | \quad PG=\{on(B,F,T^s\text{-}1)\} \\
\leftarrow \underline{\neg\delta on(B,F,T^s)} \wedge \neg ic(T^s\text{-}1) \wedge \neg\iota ic(T^s) \\
\qquad\qquad 5 \quad | \quad C=\{\leftarrow \delta on(B,F,T^s)\} \\
\leftarrow \underline{\neg ic(T^s\text{-}1)} \wedge \neg\iota ic(T^s) \\
\qquad\qquad 6 \quad | \quad PG=\{on(B,F,T^s\text{-}1), \neg ic(T^s\text{-}1)\} \\
\leftarrow \underline{\neg\iota ic(T^s)} \\
\qquad\qquad 7 \quad | \quad C=\{\leftarrow \delta on(B,F,T^s), \leftarrow \iota ic(T^s)\} \\
\qquad\qquad\quad\ [\,]
\end{array}$$

Steps 1 and 2 are SLDNF resolution steps where rules of IEM act as input clauses. At step 3 the selected literal is ιtransfer(A,y,B,t). It is current and it corresponds to a base internal event predicate. In order to make the derivation succeed we should include a ground instance of it in the input set and resolve using it as input clause. As we mentioned before, this kind of literals is included in the transition set T. Before the inclusion it is necessary to skolemise the literal obtaining ιtransfer(A,Y^s,B,T^s).

At steps 4 and 6 selected literals are on(B,F,T^s-1) and $\neg$ic(T^s-1), respectively. These are past literals. To make the derivation succeed we should include a ground instance of them in the input set and resolve using them as input clauses. Past literals are included in the previous goal set PG. In this case, skolemisation of these literals does not change them.

At steps 5 and 7 selected literals are $\neg\delta on(B,F,T^S)$ and $\neg\iota ic(T^S)$. These are current and negative literals. They must be included in the condition set C because they are conditions that the transition set T should satisfy. At the step "ensure the consistency of T", described in next section, this satisfaction will be verified and necessary additions to T, C and PG will be done if they are necessary to enforce it.

Constructive derivation description

A constructive derivation from $(G_1\ T_1\ C_1\ PG_1)$ to $(G_n\ T_n\ C_n\ PG_n)$ via a safe selection rule R, that selects literals not corresponding to evaluable predicates with priority, is a sequence:

$$(G_1\ T_1\ C_1\ PG_1), (G_2\ T_2\ C_2\ PG_2),..., (G_n\ T_n\ C_n\ PG_n)$$

such that for each i>1, G_i has the form $\leftarrow L_1 \wedge ... \wedge L_k$, $R(G_i)=L_j$ and $(G_{i+1}\ T_{i+1}\ C_{i+1}\ PG_{i+1})$ is obtained according to one of the following rules:

A1) If L_j is current, positive and it corresponds to a derived or inconsistency internal event predicate then $G_{i+1}= S$, $T_{i+1}=T_i$, $C_{i+1}= C_i$ and $PG_{i+1}=PG_i$, where S is the resolvent of some clause in IEM with G_i on the selected literal L_j.

A2) If L_j is past, positive and it corresponds to a base, derived or inconsistency predicate then $G_{i+1}= S$, $T_{i+1}=T_i$, $C_{i+1}=C_i$ and $PG_{i+1}=PG_i \cup \{s(L_j)\}$, where S is the resolvent of $s(L_j)$ with G_i on the selected literal L_j.

A3) If L_j is past, ground, negative and it corresponds to a base, derived or inconsistency predicate then $G_{i+1}= \leftarrow L_1 \wedge ... \wedge L_{j-1} \wedge L_{j+1} \wedge ... \wedge L_k$, $T_{i+1}=T_i$, $C_{i+1}=C_i$ and $PG_{i+1}=PG_i \cup \{L_j\}$.

A4) If L_j is ground and it corresponds to an evaluable predicate then $G_{i+1}= \leftarrow L_1 \wedge ... \wedge L_{j-1} \wedge L_{j+1} \wedge ... \wedge L_k$, $T_{i+1}=T_i$, $C_{i+1}=C_i$ and $PG_{i+1}=PG_i \cup \{L_j\}$.

A5) If L_j is current, positive and it corresponds to a base internal event predicate then $G_{i+1}= S$, $T_{i+1}=T_i \cup \{s(L_j)\}$, $C_{i+1}= C_i$ and $PG_{i+1}=PG_i$, where S is the resolvent of $s(L_j)$ with G_i on the selected literal L_j.

A6) If L_j is current, ground, negative and it corresponds to a base, derived or inconsistency internal event predicate "$\neg P$" then $G_{i+1}= \leftarrow L_1 \wedge ... \wedge L_{j-1} \wedge L_{j+1} \wedge ... \wedge L_k$, $T_{i+1}=T_i$, $C_{i+1}= C_i \cup \{\leftarrow P\}$ and $PG_{i+1}=PG_i$.

The step corresponding to rule A1) is an SLDNF resolution step. At steps corresponding to rules A2), A3) and A4), past and evaluable literals are added to the previous goal set PG and used as input clauses. This is done in order to have a failed derivation of IEM $\cup$ {$\leftarrow$GP$_i$} succeed. In case A5) base internal events are added to the transition set T and used as input clauses. This is also done in order to have a failed derivation of IEM $\cup$ {$\leftarrow$GP$_i$} succeed. In case A6) current negative literals are added to the condition set C, these will be conditions that T should satisfy.

There are different ways in which a constructive derivation can succeed. Each one may lead to different plans.

Let G be the current goal state and GP$_i$ a transition rule of G. T will be part of the transition from previous goal state PG to G if there exists a constructive derivation from ($\leftarrow$GP$_i$ {} {} {}) to ([] T C PG). At next step, which ensures the consistency of T with conditions in C, the rest of the transition will be determined. This step ensures consistency by additions to T, PG and C when necessary.

4.4 Ensure the consistency of T

From the constructive derivation we have obtained the transition T, the previous goal PG and the set of conditions that T must satisfy, C. However, a constructive derivation does not ensure that T satisfies conditions in C. To ensure this, SLDNF-search space for IEM $\cup$ T $\cup$ PG $\cup$ {$\leftarrow$C$_i$} must fail finitely for each C$_i$ belonging to C.

This is effected by including in PG a negated ground instance of each past selected literal corresponding to a base, derived or inconsistency predicate or selected literal corresponding to an evaluable predicate, by dropping branches where the selected literal is current, positive, corresponds to a base internal event predicate and does not unify with any fact in T, and finally, by having a constructive derivation for each current and negative selected literal. Notice that in the constructive derivation new elements may be added to T, C or PG. We will call this "consistency derivation" and a formal description of it is given in this section.

A consistency derivation can add new elements to T, C and PG in order to ensure the consistency of T. If new elements are added to T or C obtaining T' or C' then this step is reinitiated to ensure the consistency of T' and C'.

Homogenisation

In a consistency derivation, facts in T act as input clauses. Terms of these facts may be skolem constants as explained in section 4.3. At the time of the generation a skolem constant such as X^S stands for an unknown object, however, in later processing we may wish to force that object to be a particular object such as A. The only way we can do this is by adding the assumption $X^S=A$.

To be able to deal with these assumptions, facts in T with skolem constants should be homogenised before used as input clauses. A clause is homogeneous if and only if there are no constant symbols in its head atom and no variable symbol occurs more than once in its head atom. For example, $transfer(A,Y^S, Z^S, T^S)$ is not homogeneous but $transfer(x,y,z,t) \leftarrow x=A \wedge y=Y^S \wedge z=Z^S \wedge t=T^S$ is homogeneous [Esh 88].

Example

In the example of the previous section, we have obtained:

$T=\{\iota transfer(A,Y^S,B,T^S)\}$

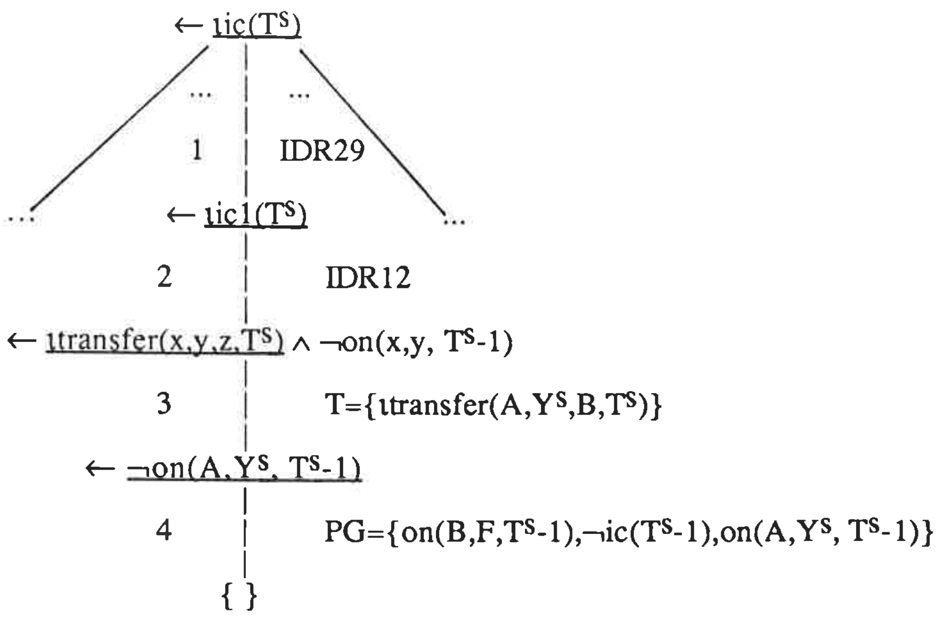
$PG=\{on(B,F,T^S-1), \neg ic(T^S-1)\}$

$C=\{\leftarrow \delta on(B,F,T^S),\leftarrow \iota ic(T^S)\}$

To ensure the consistency of T and C we must obtain a consistency derivation for each element of C: $\leftarrow \delta on(B,F,T^S)$ and $\leftarrow \iota ic(T^S)$.

$$\begin{tikzcd}
\leftarrow \underline{\delta on(B,F,T^S)} \arrow[d, dash, "1"', "IDR2"] \\
\leftarrow \underline{\iota transfer(B,F,w,T^S)} \arrow[d, dash, "2"'] \\
\{\ \}
\end{tikzcd}$$

Step 1 is an SLDNF resolution step where a rule of IEM acts as input clause. At step 2 the selected literal is current, positive and it corresponds to a base internal event but it can not be unified with any fact in T. From this, we conclude that the branch fails and it can be dropped.



$\leftarrow \underline{\iota ic(T^S)}$
...
...
1
IDR29
...
$\leftarrow \underline{\iota ic1(T^S)}$
...
2
IDR12
$\leftarrow \underline{\iota transfer(x,y,z,T^S)} \wedge \neg on(x,y, T^S\text{-}1)$
3
$T=\{\iota transfer(A,Y^S,B,T^S)\}$
$\leftarrow \underline{\neg on(A,Y^S, T^S\text{-}1)}$
4
$PG=\{on(B,F,T^S\text{-}1),\neg ic(T^S\text{-}1),on(A,Y^S, T^S\text{-}1)\}$
{ }

In this case a branch in the SLDNF-search space appears for every integrity constraint of the example but only one is shown here.

Steps 1 and 2 are SLDNF resolution steps where rules of IEM act as input clauses. Step 3 is also an SLDNF resolution step but in this case the input clause is a fact from the transition set T. At step 4 the selected literal is past. The negation of it is included in PG and thus SLDNF-search space fails finitely.

After completing the example, T and C remain unchanged and final PG is $\{on(B,F,T^S\text{-}1), \neg ic(T^S\text{-}1), on(A,Y^S,T^S\text{-}1), clear(A,T^S\text{-}1), clear(B,T^S\text{-}1), manip(A,T_0), object(B,T_0), T^S>T_0\}$. As T and C have no additions this step is finished. Otherwise, this step should be reinitiated to ensure consistency of new T and C.

Consistency derivation description

A consistency derivation from $(F_1\ T_1\ C_1\ PG_1)$ to $(F_n\ T_n\ C_n\ PG_n)$ via a safe selection rule R, that selects literals not corresponding to evaluable predicates with priority, is a sequence:

$$(F_1\ T_1\ C_1\ PG_1), (F_2\ T_2\ C_2\ PG_2),..., (F_n\ T_n\ C_n\ PG_n)$$

such that for each i>1, F_i has the form $\{\leftarrow L_1\wedge...\wedge L_k\} \cup F'_i$ and *for some* j=1...k, $(F_{i+1}\ T_{i+1}\ C_{i+1}\ PG_{i+1})$ is obtained according to one of the following rules:

B1) If L_j is current, positive and it corresponds to a derived or inconsistency internal event predicate then $F_{i+1}= S' \cup F'_i$ where S' is the set of all resolvents of clauses in IEM with $\leftarrow L_1\wedge...\wedge L_k$ on the literal L_j, $T_{i+1}=T_i$, $C_{i+1}= C_i$ and $PG_{i+1}=PG_i$.

B2) If L_j is past, ground and it corresponds to a base, derived or inconsistency predicate then $F_{i+1}=F'_i$, $T_{i+1}=T_i$, $C_{i+1}=C_i$ and $PG_{i+1}=PG_i \cup \{\neg L_j\}$.

B3) If L_j is ground and it corresponds to an evaluable predicate then $F_{i+1}=F'_i$, $T_{i+1}=T_i$, $C_{i+1}=C_i$ and $PG_{i+1}=PG_i \cup \{\neg L_j\}$.

B4) If L_j is current, positive and it corresponds to a base internal event predicate then $F_{i+1}= S' \cup F'_i$ where S' is the set of all unhomogenised resolvents of homogenised facts in T with $\leftarrow L_1 \wedge ... \wedge L_k$ on the literal L_j, and $[] \notin S'$, $T_{i+1}=T_i$, $C_{i+1}= C_i$ and $PG_{i+1}=PG_i$.

B5) If L_j is current, positive, it corresponds to a base internal event predicate and there are no homogenised facts in T that can be unified with L_j, then $F_{i+1}= F'_i$, $T_{i+1}=T_i$, $C_{i+1}= C_i$ and $PG_{i+1}=PG_i$.

B6) If L_j is current, ground, negative, it corresponds to a base, derived or inconsistency internal event predicate "$\neg P$" and there exists a constructive derivation from $(\{\leftarrow P\}\ T_i\ C_i\ PG_i)$ to $([]\ T'\ C'\ PG')$ then $F_{i+1}= F'_i$, $T_{i+1}=T'$, $C_{i+1}= C'$ and $PG_{i+1}=PG'$.

Steps corresponding to rules B1) and B4) are SLDNF resolution steps. In case B2) and B3) the negation of past or evaluable literals is added to the previous goal set PG. This is done in order to make a successful SLDNF branch fail. The branch can be dropped because failure for it is ensured. In case B5) the current branch already fails and thus it can be dropped. In case B6) the selected literal is current and negative, the current branch will be dropped if there exists a constructive derivation for the negation of the selected literal. This ensures failure for it.

Consistency derivations do not rely on the particular order in which selection rule R selects literals, since in general, all the possible ways in which a conjunction $\leftarrow L_1 \wedge ... \wedge L_k$ can fail should be explored.

Let T, C and PG be the transition, condition and previous goal sets, respectively. T and C will be consistent if there exists a consistency derivation from $(\leftarrow C_i\ T\ C\ PG)$ to $(\{\}\ T\ C\ PG')$ for each $\leftarrow C_i$ belonging to C. This can be obtained after a number of iterations of this step.

In some cases, we can have a past and not ground selected literal in a consistency derivation although that case is not defined in the rules above. A transformation is being studied that permits to incorporate them to PG. Hopefully, this case will be introduced in a next version of the approach.

4.5 Check satisfaction of PG at the initial state

This step is the same as the step described in section 4.1. At the end of each stage, the conjunction of elements obtained in PG, becomes our new goal state G for next stage so its satisfaction at the initial state must be checked as already explained.

From second stage on, as shown in section 4.3, terms in the new goal state, may be skolem constants, which stand for unknown objects. At this step they are treated as variables. It is possible that the new goal state holds at the initial state for certain values of those skolem constants.

For the rest of steps we have only shown the example at first stage because of its length. For this one, the result obtained at each stage is the following:

At first stage we obtain:

$T=\{\iota\text{transfer}(A,Y^s,B,T^s)\}$

$PG=\{\text{on}(B,F,T^s-1), \neg\text{ic}(T^s-1), \text{on}(A,Y^s,T^s-1), \text{clear}(A,T^s-1), \text{clear}(B,T^s-1), \text{manip}(A,T_0), \text{object}(B,T_0), T^s>T_0\}$

which does not hold at the initial state.

At second stage we obtain:

$T=\{\iota\text{transfer}(B,Z^s,F,T^s-1)\}$

$PG=\{\neg\text{ic}(T^s-2), \text{on}(A,Y^s,T^s-2), \text{clear}(A,T^s-2), \text{clear}(B,T^s-2), \text{manip}(A,T_0), \text{object}(B,T_0), T^s>T_0, \text{on}(B,Z^s,T^s-2), \text{clear}(F,T^s-2), \text{manip}(B,T_0), \text{object}(F,T_0), T^s-1>T_0\}$

which does not hold at the initial state.

At third stage we obtain:

$T=\{\iota\text{transfer}(A,V^s,Y^s,T^s-2), \iota\text{transfer}(X^s,B,W^s,T^s-2)\}$

$PG=\{\neg\text{ic}(T^s-3), \text{clear}(A,T^s-3), \text{manip}(A,T_0), \text{object}(B,T_0), T^s>T_0, \text{on}(B,Z^s,T^s-3), \text{clear}(F,T^s-3), \text{manip}(B,T_0), \text{object}(F,T_0), T^s-1>T_0, \text{on}(A,V^s,T^s-3), \text{on}(X^s,B,T^s-3), \text{clear}(X^s,T^s-3), \text{clear}(Y^s,T^s-3), \text{clear}(W^s,T^s-3), A\neq Y^s, X^s\neq W^s, A=X^s, V^s=B, Y^s=W^s, \text{manip}(X^s,T_0), \text{object}(Y^s,T_0), \text{object}(W^s,T_0), T^s-2>T_0, A\neq W^s, B\neq X^s, F\neq Y^s, F\neq W^s\}$

which holds at the initial state for $X^s=A$, $V^s=B$, $Z^s=C$, $W^s=G$, $Y^s=G$ and $T^s-3=T_0$.

From above, the plan obtained is:

$P=\{\iota\text{transfer}(A,B,G,T_0+1), \iota\text{transfer}(B,C,F,T_0+2), \iota\text{transfer}(A,G,B,T_0+3)\}$

5. CONCLUSIONS

We have presented our approach to the problem of validation of DCMs. The approach generates a plan in n stages obtaining at each of them the events of the plan that occur at each time point. At each stage we use the IEM which explicitly defines insertions and deletions of information induced by previous occurrence or absence of external events. An extension of SLDNF procedure has then been used to obtain sequences of external events that lead to the goal state. The generated plan satisfies integrity constraints if the goal state is adequately formulated.

As we have already mentioned, our problem is strongly related to that of plan formation. Kowalski shows in [Kow 79] how to deal with this problem in logic and using the operational approach. In our case, we also deal with the problem in logic but we take the deductive approach for the conceptual modelling of information systems. In the operational approach, external events or actions are defined in terms of its preconditions and its postconditions (informations added or deleted by the action). Due to this, in Kowalski's method an additional frame axiom is needed to deal with informations that were true in the previous state and have not been deleted. In the deductive approach, the truth value of informations is given by its deduction rules and the frame axiom is implicit in them.

On the other hand, in Kowalski's method, it is possible to use simple SLDNF resolution to generate the plan because preconditions of actions are explicit and only one action may be performed in the transition from one state to another. In the deductive approach, preconditions are not explicit and it is permitted that several actions occur at the same time.

Kowalski presents two ways of generating the plan: bottom-up, from the initial state to the goal state, or top-down, from the goal state to the initial state. From this point of view, our approach is top-down.

A direction of future work is to study optimizations in order to implement the approach efficiently.

ACKNOWLEDGEMENTS

We would like to thank Enric Mayol, Antoni Olivé, Joan Antoni Pastor, Carme Quer, Maria Ribera Sancho, Jaume Sistac, Ernest Teniente, Carme Torras and Toni Urpí for many useful comments and discussions.

REFERENCES

[BuO 86] Bubenko, J.; Olivé, A. "Dynamic or temporal modelling?. An illustrative comparison", SYSLAB Working Paper No.117, University of Stockholm, 1986.

[Dec 89] Decker, H. "Drawing updates from derivations" Proc. of the ICDT 90, Springer 1990, pp. 437-455.

[Esh 88] Eshghi, K. "Abductive Planning with Event Calculus", Proc. of the Fifth International Logic Programming Conference, MIT Press, 1988.

[FuC 90] Furtado, A.L.; Casanova, M.A. "Plan and schedule generation over temporal databases", Proc. of the 9th International Conference on Entity-Relationship Approach, Lausanne, 1990, pp. 235-248.

[FuM 84] Furtado, A.L.; Moura, C.M.O. "Expert helpers to data-based Information Systems", First International Workshop on Expert Database Systems, 1984.

[GKB 82] Gustaffson, M.; Karlsson, T.; Bubenko, J. "A declarative approach to conceptual information modelling", In [OSV 82], pp. 93-142.

[Gri 82] van Griethuysen, J.J. (Ed.) "Concepts and terminology for the conceptual schema and the information base", ISO/TC97/SC5/WG3/, March 1982.

[KaM 90] Kakas, A.C.; Mancarella, P. " Database Updates Through Abduction", Proc. of the 16th VLDB, Brisbane, pp. 650-661.

[Kow 79] Kowalski, R.A. "Logic for problem solving", North-Holland, 1979.

[Oli 82] Olivé, A. "DADES: A methodology for specification and design of information systems", In [OSV 82], pp. 285-334.

[Oli 86] Olivé, A. "A comparison of the operational and deductive approaches to conceptual information systems modelling", Proc. IFIP-86, Dublín, pp. 91-96.

[Oli 89] Olivé, A. "On the design and implementation of information systems from deductive conceptual models", Proc. of the 15th VLDB, Amsterdam, 1989, pp. 3-11.

[OSV 82] Olle, T.W.; Sol, H.G.; Verrijn-Stuart, A.A. (Eds.) "Information systems design methodologies: A comparative review", North-Holland, 1982.

[San 90] Sancho, M.R. "Deriving an internal events model from a deductive conceptual model", Proc. of the International Workshop on the Deductive Approach to Information Systems and Databases, S'Agaró (Catalonia), 1990, pp. 73-92.

[TeO 91] Teniente, E; Olivé, A. "The Events Method for View Updating in Deductive Databases", Internal Report.

[Vef 85] Veloso, P.A.S.; Furtado, A.L. "Towards simpler and yet complete formal specifications", Proc. of the IFIP Working Conference on Theoretical and Formal Aspects of Information Systems, 1985, pp. 175-189.