Handling Conceptual Model Validation by Planning

Dolors Costal, Ernest Teniente, Toni Urpí and Carles Farré

Universitat Politècnica de Catalunya
LSI, Facultat d'Informàtica
Pau Gargallo 5. E-08028 Barcelona -- Catalonia
e-mail: [dolors | teniente | urpi]@lsi.upc.es, farre@goliat.upc.es

Abstract. An important amount of research has been devoted to conceptual model validation, that is, to check whether a conceptual model correctly and adequately describes the users' intended needs and requirements. In this paper we present a new approach to model validation. We define a set of desirable properties that a conceptual model should satisfy and we show how the accomplishment of all these properties can be checked in a uniform way by means of planning. Our approach is independent of any particular planning method and it extends the facilities of the methods developed so far.

1 Introduction

This paper describes an approach for validating conceptual models of information systems. By validation we mean the process of checking whether a conceptual model correctly and adequately describes the users' intended needs and requirements [ABC82]. This is one of the most important and crucial problems in information systems engineering. Indeed, detecting possible problems during information systems specification time will prevent possible errors during the following stages.

The validation task is not fully formalizable and it is based on intuition. Then, it is desirable to provide the designer with the maximum set of tools that assist him in the validation process [Bub86]. Among the support capabilities that have been proposed there are: infological simulation [VF85, CO92], paraphrasing specifications in natural language [RP92, Dal92], generation of abstractions and abstracts of specifications [JC92], animation and symbolic execution [LL93], explanation generation [GW93, OS95] and semantic prototyping [LTP91, LK93].

One of the proposed approaches for validating conceptual models consists of defining a set of desirable properties that the model should satisfy and to provide methods for checking these properties [Kun84, Kun85]. However, this proposal has several drawbacks. First, it deals only with a reduced set of properties. Second, it does not take into account the presence of the operations of the conceptual model in both the definition and the validation of the properties. Finally, it lacks of a uniform treatment of the properties in the sense that they are checked using different techniques. The work that we present in this paper extends previous research in this direction.

In our approach, the checking of the considered properties will be performed by using plan generation techniques. We define for each property (e.g. satisfiability of a conceptual model, redundancy of integrity constraints or executability of an operation) an initial state and a final state such that if there exists a plan, that is, a sequence of operations able to perform the transition between both states, the conceptual model satisfies that property. The use of planning allows us to check the whole set of properties in a uniform way. Moreover, it allows us to easily consider additional

desirable properties and to guarantee that their checking is performed correctly, due to the existence of adequate planning methods.

We would like to point out that the generality of our approach allows us to be independent of the concrete planning method used for checking the properties. This facilitates the application of the improvements in the planning research area to our approach.

Furtado et al. [VF85] proposed a concrete planning method for reasoning about conceptual models. In their approach, the designer is provided with a tool that obtains a sequence of operations able to perform the transition between a given initial and target states. Therefore, the success of validation relies only on the designer's intuition and experience when defining the initial and target states. Moreover, Furtado et al. do not consider the integrity constraints when reasoning about conceptual models.

Recently, Feenstra and Wieringa [FW95] have also outlined a method for reasoning about conceptual models specified in a language less expressive than ours which does not consider derived predicates, in a similar way than the proposed by Furtado et al.. They propose a system able to answer reachability queries which permits to express some properties similar to ours. Their method is based on extending planning techniques with a satisfiability tester [BDM88]. However, as we will show in this paper, these properties can be checked by using some of the current planning methods.

This paper extends our previous work on validation in the context of databases [DTU96], which focused on the use of view updating for validating a database schema, without taking the predefined transactions or operations into account. It also extends our work reported in [CO92], where we proposed a concrete planning method for explaining the reachability of a final state, departing from a given initial state. As for [VF85], the success of validation relied only on the designer's intuition and experience when defining the initial and target states.

This paper is organized as follows. Next section reviews basic concepts of usual conceptual models, which are based on the concept of operation. Section 3 reviews the main planning concepts and identifies the features of planning methods able to handle the validation tasks considered in this paper. Section 4 defines a set of desirable properties that a conceptual model should satisfy and shows how the checking of these properties can be handled using planning. Section 5 illustrates how to apply our approach by using two existing planning methods. Finally, section 6 presents our conclusions and points out further work.

2 Conceptual models

In this section we review usual conceptual models of information systems and we introduce our notation for describing them. Most conceptual models are based on the concept of operation. We want to be general, and therefore we use a simple formalism, based on the concept of operation, and easily adaptable to many conceptual modelling languages.

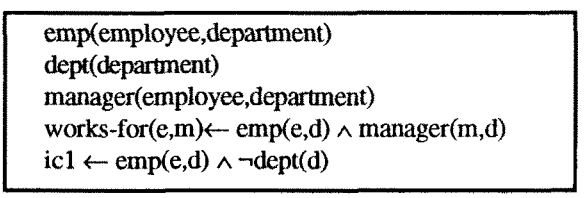
Each conceptual modelling language provides a set of concepts and syntactic features to define the structure of the Information Base: the Information Base schema. In general, we may assume that the Information Base (IB) consists of two parts: *Base* and

Derived. The Base part includes all facts that are inserted and deleted directly by the operations, while the Derived part includes all facts that are derived from base and/or derived facts, by means of some kind of deductive rules.

We will assume the IB contains facts of a given set of *predicates* . Each predicate consists of a name and a set of arguments. Facts of base predicates are updated by operations while the existence of facts of derived predicates is defined declaratively by *deductive rules*. We will use the clausal form of logic, augmented with negation, to define deductive rules.

Many conceptual modelling languages provide some constructs for the definition of integrity constraints. For the sake of uniformity, we define the constraints in denial form by means of integrity rules which have the same form as the deductive rules. Any integrity constraint can be represented in this form as described in [LT84].

Below, we show an example of an IB schema:



There is a single derived predicate: works-for. Its deductive rule indicates that an employee E works for another employee M if E is an employee of a department D and M is the manager of that department. Note that we also include an integrity rule. It states that all employees must belong to an existent department.

It can be seen that there is a straightforward correspondence between our IB schema and that of ER [Che76], SDM [HM81], NIAM [NH89], and many others. Note that not all of them include a derived part in the IB. Our model is even adaptable to languages based on the relational data model. In such case, views are derived predicates and their definition is a deductive rule.

A conceptual modelling language has also to provide the definition of the transitions between different states of the Information Base: the transition schema. Most conceptual modelling languages define the transition schema in terms of operations. Application of an operation induces a transition from the current state of the IB to a new state.

We describe operations by: its name, its parameters, a precondition that must hold in the state of the IB in which the operation is applied, facts to be added to the IB by the operation and facts to be deleted from the IB by the operation.

Preconditions are expressed as first order formulas. In order to guarantee domain independence, range-restrictedness of the formulas is required. We use the common definition of range-restrictedness as introduced in [Nic82]. Additions and deletions can be expressed as sets of base facts.

In this paper, we assume that the specified precondition guarantees that facts to be deleted hold in the IB state in which the operation is applied and facts to be added do not hold in that IB state. However, our approach could also be applied if this requirement was not considered. Below, we show the transition schema for the IB schema described.

New-dept(D)	**Rem-dept(D)**
prec: ¬ dept(D)	prec: D≠Staff ∧ dept(D)
add: {dept(D)}	add: {}
del: {}	del: {dept(D)}
Hire(E,D)	**Fire(E)**
prec: ¬ ∃d' emp(E,d')	prec:∃d emp(E,d)∧¬manager(E,d)
add: {emp(E,D)}	add: {}
del: {}	del: {emp(E,d)}
New-manager(E,D)	**Rem-mgr(E)**
prec: ¬∃d' manager(E,d')	prec:∃dmanager(E,d)
add: {manager(E,D)}	add: {}
del: {}	del: {manager(E,d)}

The execution of an operation is *valid* only if its precondition holds in the IB state in which the operation is applied and the resulting IB state obtained after its execution is consistent, i.e., integrity constraints are satisfied in it. Otherwise, the operation can not be performed.

For example, the execution of operation Hire(John,Marketing) is not valid when applied to an IB state where base fact dept(Marketing) does not hold. In this case, emp(John,Marketing) would hold in the resulting state and dept(Marketing) would not. Thus, integrity constraint ic1 would be violated in the resulting state of the IB and this resulting state would be inconsistent.

Similarly, a sequence of operations is valid only if the execution of each operation of the sequence is valid. Moreover, an IB state is *reachable* only if it can be obtained by a valid sequence of operations.

In our example, the IB state: {dept(Marketing), emp(John,Marketing)} is reachable due to the following sequence of operations: New-dept(Marketing), Hire(John,Marketing)

As it happens with our IB schema, there is a straightforward correspondence beetween our transition schema and the transition schema of many conceptual modelling languages, such as that of Taxis [MBW80] and the one introduced by Furtado et al. in [VF85].

3 Planning

In this section we present classical planning problems and we identify the features of planning methods able to handle the validation tasks considered in this paper.

A classical planning problem can be stated as follows: given a description of the world or domain of the problem, given an initial state of that world and given a goal to accomplish at a final state, obtain one or several plans able to perform the transition between the initial state and a final state in which the desired goal holds. The obtained plans have to be consistent with respect to the world description. Note that, in particular, the obtained plan may be empty which means that the initial state already satisfies the goal.

A planning method has to provide both a way to *represent* a planning problem and a mechanism to *generate* plans for the planning problems it is able to represent.

The representation used by a planning method is important because it characterizes the planning problems it is able to solve. Actually, the representation of a planning problem is strongly related with conceptual modelling of information systems. In both cases, it is necessary to represent static and dynamic knowledge about a domain that evolves through several situations or states.

In order to use a planning method for the validation of the conceptual models described in section 2, it must provide a representation of a planning problem in terms of those conceptual models either directly or through an automatic transformation.

Using this representation the planning problem can be formulated more concretely as: given an IB schema and a transition schema, given an initial state of the IB expressed as a set of base facts[1] and given a goal expressed as a range-restricted first order formula to accomplish at a final state of the IB, obtain one or several valid sequences of operations able to perform the transition between the initial state of the IB and a final state of the IB where the desired goal holds.

For example: given the planning problem domain described by the IB schema and transition schema examples of section 2, given the initial state: {dept(Marketing), emp(John,Marketing)} and given the goal: $\exists$d emp(Peter,d) $\wedge$ $\exists$e works-for(e,Peter), a possible plan to obtain is the valid sequence of operations: Hire(Peter,Marketing), New-mgr(Peter,Marketing).

We also have a requirement on the plan generation mechanism. It must obtain at least a solution plan for a given planning problem if any solution plan for the planning problem exists. The need of this requirement will become clear in section 4 when defining the properties to validate in terms of planning.

4 Using Planning for Validating Conceptual Models

In this section we define a set of desirable properties that a conceptual model should satisfy and we show how the checking of these properties can be handled by using planning. The main idea is to define for each property an initial state and a goal to accomplish at a final state such that if there exists a sequence of operations able to achieve the goal departing from the initial state, the conceptual model satisfies that property. If a property is not satisfied, the conceptual model is ill-specified and its specification should be changed. In general, this may be due to the specification of the deductive rules, integrity constraints, operations or even due to a combination of them.

In sections 4.1 and 4.2 we deal respectively with the properties related to the validation of the IB schema and to the validation of the operations.

4.1 Checking of Properties of the IB Schema

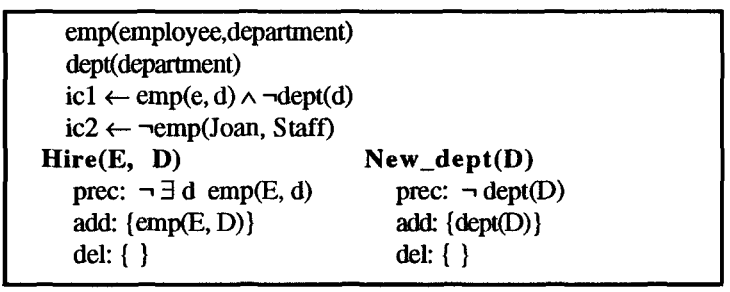
In this section we will define three properties related to the Information Base schema: satisfiability, liveliness of a predicate and redundancy of a constraint. We will

[1] Note that we only need the base facts of the IB to describe the initial state because the derived facts may be derived using the deductive rules. In particular, this set of base facts may be empty and we will call empty initial state its corresponding initial state

also show how these properties can be uniformly handled using planning. As a related work, we would like to mention the methods for handling satisfiability proposed in [Kun84, Kun85] and [Lun82]. These methods are only concerned with finding a model that satisfies all the integrity constraints since they do not take the operations into account in both the definition and the verification of the property. In the database field, three different methods [BDM88, LMSS93, GSUW94] have been proposed that handle respectively satisfiability, liveliness and redundancy. Besides the non uniformity of the way in which these properties are handled, again the main difference relies on the fact that these methods do not consider the operations.

4.1.1 Satisfiability

A fundamental property that a conceptual model should satisfy is satisfiability. Intuitively, a conceptual model is satisfiable if there exists a state of the IB satisfying all the integrity constraints which can be obtained by a sequence of operations. If there is not such a state then the entire conceptual model becomes useless since no operation can be executed. To illustrate this property, consider a subset of the conceptual model defined in section 2 with the additional integrity constraint ic2, stating that it must exist the employee Joan assigned to the department Staff:



This conceptual model is clearly unsatisfiable. Satisfaction of integrity constraint ic2 would require a state in which at least emp(Joan, Staff) was true, which could be achievable by the operation Hire(Joan, Staff), however the execution of this operation would violate ic1 since the department Staff would not exist. Thus, it would be necessary to execute previously the operation New_dept(Staff) that can not be executed since this execution would violate ic2.

Satisfiability in this example could be achieved, for instance, by removing integrity constraint ic2, in which case the conceptual model becomes satisfiable since the empty state satisfies all the integrity constraints. Satisfiability could also be achieved by proposing a new transaction that inserts at the same time an employee and a department.

More precisely, a conceptual model M is *satisfiable* if there exists a reachable state of the Information Base.

Satisfiability of a conceptual model M is equivalent to the planning problem of finding a plan for the goal TRUE where the description of the world is given by the conceptual model M and the initial state is empty. The tautology TRUE is the least restrictive possible goal. Thus, if it does not exist a plan for that goal, the integrity

constraints can not be satisfied and therefore the conceptual model is unsatisfiable. On the other hand, if there exists at least a plan for that goal, then there exists a valid sequence of operations that leads the Information Base to an state in which all the integrity constraints are satisfied and therefore the conceptual model is satisfiable.

Note that, if the operations of the conceptual model are not taken into account, the state {emp(Joan,Staff), dept(Staff)}, for instance, would satisfy all the integrity constraints and therefore the previous conceptual model would be satisfiable according to the definition of satisfiability considered in [Lun82, Kun84, Kun85]. However, as we have seen, this state is not reachable by the operations of the model.

4.1.2 Liveliness of a Predicate

A desirable property for base and derived predicates is that they can have non empty extensions. Effectively, a predicate that has an empty extension in each reachable state of the IB is clearly a useless predicate and probably the conceptual model in which it is defined is ill-specified. As an illustrative example, consider again a subset of the conceptual model defined in section 2:

emp(employee,department)
dept(department)
ic1 $\leftarrow$ emp(e, d) $\wedge$ $\neg$dept(d)
Hire(E, D)
prec: $\neg \exists$ d emp(E, d)
add: {emp(E, D)}
del: { }

Clearly, both predicates dept and emp are not lively. The first one, dept, is not lively since there is no operation to insert a department and consequently it will have an empty extension in each possible state. The second one, emp, is also not lively since even though there is an operation to insert an employee this operation may never satisfy the integrity constraint ic1 that requires the existence of the department when inserting an employee. If we assume a new operation:

New-dept(D)
prec: $\neg$ dept(D)
add: {dept(D)}
del: {}

then both predicates become lively: predicate dept due to the presence of this new operation and predicate emp due to the fact that the sequence of operations: New_dept(Dept), Hire(Emp, Dept) leads to an state of the Information Base in which emp has the fact emp(Emp,Dept). Note that the inverse sequence Hire(Emp, Dept), New-dept(Dept) is not a valid sequence since the execution of the first operation would violate ic1.

More precisely, a predicate p is *lively* in a conceptual model M if there is a reachable state of the Information Base in which at least one fact about p is true.

The problem of finding out whether a predicate p(x)[2] is lively in a conceptual model M is equivalent to the planning problem of finding a plan for the goal $\exists \mathbf{x}\ p(\mathbf{x})$ where the description of the world is given by the conceptual model M and the initial state is empty. If there is at least a plan for such a goal then the plan will contain a valid sequence of operations that leads to a reachable state in which p has a non empty extension and thus, predicate p is lively. Otherwise p is not lively.

4.1.3 Redundancy of a Constraint

Intuitively, a constraint is redundant if integrity does not depend on it. Clearly, such a redundancy should be detected by the validation process and the designer should be informed in order that s/he could take the opportune decisions. Redundancy of a constraint may be due to several reasons. For instance, a constraint may be a tautology, it may be redundant because it is enforced by the specifications of the operations, by other integrity constraints or due to a combination of these reasons. To illustrate this property, consider a subset of the conceptual model defined in section 2 with two additional integrity constraints:

emp(employee,department)
dept(department)
ic1 $\leftarrow$ emp(e, d) $\wedge$ $\neg$dept(d)
ic2 $\leftarrow$ emp(Joan, d) $\wedge$ $\neg$dept(d)
ic3 $\leftarrow$ emp(e, d1) $\wedge$ emp(e, d2) $\wedge$ d1$\neq$d2

Hire(E, D)	**New-dept(D)**
prec: $\neg \exists$ d emp(E, d)	prec: $\neg$ dept(D)
add: {emp(E, D)}	add: {dept(D)}
del: { }	del: { }

Integrity constraint ic2 is redundant, since it is entailed by ic1 and therefore it can not be violated without violating ic1. Moreover, integrity constraint ic3 is redundant since the precondition of the operation Hire forbids to have an employee assigned to more than one department. Thus, the integrity of the conceptual model does not depend on ic2 neither ic3, in the sense that they could be removed and the resulting conceptual model would admit the same reachable states than the original one.

More precisely, let M be a conceptual model and let I be an integrity constraint in M, I is *redundant* if it is satisfied in each reachable state of M - {I}.

Satisfiability and liveliness can be checked by showing the existence of some reachable state. As opposed to that, the definition of redundancy of a constraint requires something for *each* state. Hence, redundancy of a constraint is best checked by verifying or falsifying the lack of redundancy, which can be done by attempting to construct *one* state which would show that the constraint under investigation can be violated and hence that is not redundant. Thus, the non redundancy of a constraint I of the form ic $\leftarrow$ Body in a conceptual model M is checked by showing the existence of a plan for the goal $\exists \mathbf{x}$ Body, where **x** are the variables in Body, with M - {I} as a

[2] Note that here and in the following bold lowercase letters stand for vectors of variables

description of the world and with an empty initial state. If it does not exist such a plan then the integrity constraint is redundant.

In our example it does not exist any plan for the goal $\exists$ d emp(Joan, d) $\wedge$ $\neg$dept(d) in the world corresponding to the original conceptual model except ic2. The same happens with goal $\exists$ e,d1,d2 emp(e,d1) $\wedge$ emp(e,d2) $\wedge$ d1$\neq$d2 in the world corresponding to the original conceptual model except ic3. Thus, both ic2 and ic3 are redundant. Instead, since the plan Hire(Peter,Marketing) is a possible plan for the goal $\exists$ e,d emp(e, d) $\wedge$ $\neg$dept(d) in the world corresponding to the original conceptual model except ic1, ic1 is not redundant.

4.2 Checking of Properties of the Operation Specifications

In addition to the checking of properties of the Information Base, the designer may also be interested in the checking properties of the operations of the conceptual model. For instance, s/he could be interested on checking whether the precondition of each operation is correctly specified or whether it will be possible to execute the specified operations at execution time. These properties, i. e. the applicability of an operation and the executability of an operation, are defined in this section.

Similar properties to our applicability and executability of an operation were defined in [Kun84, Kun85]. Besides the non uniformity of the way in which these properties are handled, again the main difference relies on the fact this approach does not take account of the presence of the other operations in both the definition and the verification of the properties related to an operation.

4.2.1 Applicability of an Operation

Intuitively, an operation is applicable if there exists a reachable state where its precondition holds. If an operation is not applicable, it will never be possible to execute it because its precondition may never be true. As an example, consider the following conceptual model:

emp(employee, department)
manager(employee, department)
dept(department)
has-emp(d) $\leftarrow$ emp(e,d)
has-emp(d) $\leftarrow$ manager(e,d)
has-mgr(d) $\leftarrow$ manager(e,d)
ic1 $\leftarrow$ dept(d) $\wedge$ $\neg$ has-mgr(d)

Hire(E,D)	**New-mgr(E,D)**
prec: $\neg$ $\exists$d' emp(E,d')	prec: $\neg$ $\exists$d' manager(E,d')
add: {emp(E,D)}	add: {manager(E,D)}
del: {}	del: {}
New-dept(D)	**Rem-dept(D)**
prec: $\neg$dept(D)	prec: dept(D) $\wedge$ $\neg$has-emp(D)
add: {dept(D)}	add: {}
del: {}	del: {dept(D)}

At a first glance, it may seem that all the operations of this conceptual model are applicable. However, consider the operation of removing a department: Rem-dept(D). The satisfaction of the first part of the precondition, dept(D), requires the previous execution of the operation New-dept(D). Moreover, because of the integrity constraint ic1, the operation New-dept(D) can only be executed if department D has some manager, which in turn requires the previous execution of the operation New-mgr(E,D) to add the information manager(E,D) to the IB. By the second derivation rule, manager(E,D) implies has-emp(D) and, since there is no possible way to remove this information, it cannot be achieved a state where the precondition of the operation Rem-dept(D) holds. Therefore, we have that this operation is not applicable. Several ways of making operation Rem-dept(D) applicable exist, for instance by changing its precondition, by changing the deductive rules or integrity constraints of the IB or by considering additional operations.

More precisely, an operation Op is *applicable* if there is a reachable state S where, for some instantiation of the parameters of Op, the precondition of Op holds in S.

As with the other properties, the problem of deciding whether an operation is applicable may also be handled by means of planning. In this case, we have that an operation Op, with parameters **x** and precondition Prec, will be applicable if there exists a valid sequence of operations that, starting from an empty initial state, leads to a final state of the IB that satisfies the goal: $\exists \mathbf{x}$ Prec. In the previous example, since the operation Rem-dept(D) is not applicable, no plan exists that satisfies the goal: $\exists$d dept(d) $\wedge \neg$has-emp(d). However if, for instance, we remove the second deductive rule, the following sequence of operations would be a solution plan: New-mgr(Peter,Sales), New-dept(Sales). Therefore, the operation Rem-dept(D) would be applicable.

4.2.2 Executability of an Operation

Even though an operation is applicable, it may be the case that it may never be executed because the result of applying this operation to each of the states that satisfy its precondition leads to an inconsistent resulting IB state. Detection of non-executable operations is also an important task in conceptual model validation. This is illustrated in the following example.

manager(employee, department)		
dept(department)		
has-mgr(d) ← manager(e,d)		
ic1 ← dept(d) ∧ ¬ has-mgr(d)		
New-dept(D)	**New-mgr(E,D)**	**Rem-mgr(E)**
prec: ¬dept(D)	prec: ¬ ∃d' manager(E,d') ∧ ¬ ∃e' manager(e',D)	prec:∃d manager(E,d) ∧ dept(d)
add: {dept(D)}	add:{manager(E,D)}	add: {}
del: {}	del: {}	del: {manager(E,d)}

Consider the previous conceptual model, where all the operations are a subset of the operations of the example considered in section 2, except that we have slightly changed

the precondition of the operation New-mgr(E,D) by requiring also that it may only be applied if D has no manager, and the precondition of the operation Rem-mgr(E) by requiring that E must be the manager of some known department.

Rem-mgr(E) is an applicable operation because there exists a reachable state, obtained by the execution of the operations: New-mgr(E,D), New-dept(D), that satisfies its precondition. However, it will never be possible to apply it because its execution always leads to an inconsistent IB state. It is not difficult to see that the reason why Rem-mgr(E) is not executable lies on the fact that the integrity constraint ic1 requires all departments to have a manager and that the precondition for applying New-mgr(E,D) requires that no manager exists for D. Therefore, in this conceptual model a department may never have two managers and, then, the application of the operation Rem-mgr(E) will always result in a state that violates ic1.

More precisely, an operation Op is *executable* if there exists a valid sequence of operations that contains Op.

Checking the executability of an operation Op by means of planning has to deal with the problem of finding a plan that contains Op and that satisfies certain conditions of the IB. Unfortunately, most of the known planning methods do not allow the specification of conditions that the plan to obtain must satisfy. However, if the IB keeps trace of the executions of Op it is also possible to check the executability of an operation Op with usual planning methods. In conceptual models, this can be done by including in the specification of Op a new information to be added to the IB: Executed-Op, where Executed-Op is a distinguished predicate which does not occur elsewhere in the IB. Then, the executability of the operation Op is checked by showing the existence of a valid plan for the goal Executed-Op, starting from the empty initial state. If some plan exists, it must necessarily contain Op since the execution of this operation is the only possible way to add Executed-Op to the IB, and thus Op will be executable; otherwise it will not.

In the previous example, the operation Rem-mgr(E) should be described as:

Rem-mgr(E)
prec: $\exists$d manager(E,d) $\wedge$ dept(d)
add: {Executed-Rem-mgr}
del: {manager(E,d)}

and since no plan exists that satisfies the goal Executed-Rem-mgr the operation Rem-mgr(E) is not executable. Note that if we remove the second part of the precondition of New-mgr(E,D) then Rem-mgr(E) would be executable since: New-mgr(Sue,Sales), New-dept(Sales), New-mgr(Mary, Sales), Rem-mgr(Sue), would be a valid plan for the goal Executed-Rem-mgr.

5 Using Planning Methods for Conceptual Model Validation

In this section we illustrate our approach to tackle conceptual model validation by using some of the actual planning methods.

In order to use a planning method for the validation of the conceptual models described in section 2 (operational conceptual models in the following), it must provide a representation of a planning problem in terms of those operational

conceptual models. This representation may be provided either directly or through an automatic transformation.

We will consider two planning methods: CDP [DMMP91] and our planning method [CO92, Cos95]. CDP will illustrate the case in which the required representation is directly provided while our planning method will illustrate the case in which the representation is provided by using an automatic transformation. This automatic transformation will show that a planning method using a representation not based on the concept of operation but only based on deductive rules can also be used for our validation tasks.

We also have a requirement on the plan generation mechanism. It must obtain at least a solution plan for a given planning problem if any solution plan for the planning problem exists. Note that this requirement is less restrictive than obtaining the complete set of solution plans. Both CDP and our planning method fulfill this requirement.

5.1 CDP [DMMP91]

CDP uses deductive databases for modelling states of the planning problem world and it uses operation specifications to model transitions between those states.

CDP obtains a complete set of correct minimal solution plans for a given planning problem. A solution plan is minimal if none of its subsequences of operations is also a solution plan. Therefore, CDP obtains at least a solution plan for a given planning problem if any solution plan for the planning problem exists.

Components of planning problems represented by CDP have a direct correspondence to a planning problem represented in terms of our operational conceptual models. We will illustrate this direct correspondence by using the example presented in section 4.2.1 to check the applicability of operation Rem-dept .

The kind of deductive databases treated by CDP can be viewed as an IB of an operational conceptual model. The IB schema with some syntactic changes will be a deductive database schema of CDP. In our example, rules corresponding to predicate has-emp after the syntactic changes will be: $\forall$e,d emp(e,d)$\rightarrow$has-emp(d), $\forall$e,d manager(e,d)$\rightarrow$has-emp(d)

Operation specifications consist of: a head, $op(x_1,...,x_n)$, where op is the operator name and the parameters $x_1,...,x_n$ are variables, a precondition and a postcondition. Preconditions are range-restricted first order formulas which define the database state where the operation is applicable. Postconditions describe the additions and deletions of base facts that must be performed to obtain the operation resultant database state. Additions are described by positive literals and deletions are described by negative literals. Therefore, a simple rewriting will turn our operations into CDP operations. In our example, the rewriting of operations New-dept(D) and Rem-dept(D) is:

New-dept(D)	**Rem-dept(D)**
prec: $\neg$dept(D)	prec: dept(D) $\wedge$ $\neg$has-emp(D)
post: dept(D)	post: $\neg$ dept(D)

The initial state of the planning problem world is described as the deductive database initial state. This deductive database initial state can be seen as the initial state of the

IB of an operational conceptual model. In our example, the empty initial state will correspond to a deductive database with an empty set of base facts.

The planning problem goal G is defined as a set of range-restricted first order formulas. For uniformity reasons, it is assumed that a goal G implicitly introduces an operation *goal()*. The precondition of *goal()* is G and its postcondition is empty. Again, a simple rewriting will turn our goal into the CDP operation *goal()*. In our example, for applicability of operation Rem-dept we will have:

goal()
prec: $\exists d\ dept(d) \wedge \neg has\text{-}emp(d)$
post:

A plan for a planning problem is described as a sequence of operations where the last operation is *goal()*. A plan is a solution for a planning problem if for all operations in the plan: its precondition is satisfied at the database state where the operation is applied and the database state resulting from the operation application is consistent, that is, integrity constraints are satisfied in it. Therefore, this sequence of operators will be a solution plan for the planning problem. In our example, no sequence is obtained which shows that operation Rem-dept is not applicable.

5.2 Our Planning Method [CO92, Cos95]

In [CO92, Cos95] we defined a planning method which uses a representation of the planning problem in terms of deductive conceptual models. Deductive conceptual models differ mainly from usual operational conceptual models of information systems in that they do not use the concept of operation and the transitions of the IB schema are entirely described in terms of deductive rules.

Our planning method obtains a complete set of correct minimal solution plans for a given planning problem. A solution plan is minimal if none of its subsets is also a solution plan. Therefore, our planning method obtains at least a solution plan for a given planning problem if any solution plan for the planning problem exists.

In deductive conceptual models time is a key concept. The passing of time is considered as an external event (or operation occurrence) as any other entry to the system. Consequently, there exists a state of the IB for each time point. Each possible information has an associated time point. The Information Base (IB) consists of two parts: *Base* and *Derived*. In deductive conceptual models the Base part at a time point t corresponds to the information of all the external events (or operation instances) that have occurred at time point t. As the passing of time is an external event, the Base part contains facts of an standard predicate time(t). A fact time(T) holds if T belongs to the life span of the system. The rest of the information, that is, the Derived part, can be deduced from the Base part by means of deductive rules. At a given time point t, the IB contains all the informations known until t. In deductive conceptual models the transition schema is embedded in the deductive rules.

In the following, we describe how to transform each of the components of an operational conceptual model in order to obtain a deductive conceptual model. We will illustrate this transformation by using the example presented in section 2.

For each operation Op(**x**) we define a corresponding base predicate op(**x**,t) of the

deductive conceptual model. Note that we have an additional time term t to indicate its time of occurrence. In our example, we will have the following five base predicates: new-dept(d,t), rem-dept(d,t), hire(e,d,t), fire(e,t), new-manager(e,d,t), rem-mgr(e,t). The knowledge represented in the body of operations will be transformed in deductive rules as we will see in the following paragraphs.

For each base predicate p(x) we define a corresponding derived predicate p(x,t) of the deductive conceptual model. Again, the additional time term t indicates the time of occurrence of the associated information. In our example, we will have that emp(e,d,t), dept(d,t) and manager(e,d,t) are derived predicates.

We have to define deductive rules for these derived predicates. These deductive rules will represent the knowledge that in our previous operational conceptual models was described by operations. For each base predicate p(x) of our previous model whose facts are added by operations $a_1,...,a_n$ and deleted by operations $d_1,...,d_m$ we will define the following deductive rules in our deductive conceptual model: add-p(x,t)$\leftarrow$ $a_1(y_1,t) \wedge$ prec-a_1, ..., add-p(x,t)$\leftarrow$ $a_n(y_n,t) \wedge$ prec-a_n, del-p(x,t)$\leftarrow$ $d_1(y_1,t) \wedge$ prec-d_1, ..., del-p(x,t)$\leftarrow$ $d_m(y_m,t) \wedge$ prec-d_m, p(x,t)$\leftarrow$ add-p(x,t1) $\wedge$ t1$\leq$t $\wedge \neg \exists$t2 del-p(x,t2) $\wedge$ t2>t1 $\wedge$ t2$\leq$t.

We have defined auxiliary derived predicates add-p and del-p which indicate the addition and deletion of predicate p by an operation, respectively. Rules for these predicates permit to deduce a fact of them at a time point T if the operation that performs the addition or deletion has occurred at time T and the precondition of the operation held at the previous time T-1. We assume that if F is the precondition of the operation Op then prec-op stands for the formula F with the addition of a time term t-1 to each predicate of the formula. Then, it is easy to define a rule for predicate p which states that a fact of this predicate holds at a time T if it has been added at a time T1 before T and has not been deleted at a time T2 between T1 and T.

In our example we will have the following rules for predicate emp(e,d,t):

add-emp(e,d,t)$\leftarrow$ hire(e,d,t) $\wedge \neg \exists$ d' emp(e,d',t-1)

del-emp(e,d,t)$\leftarrow$ fire(e,t) $\wedge \exists$d emp(e,d,t-1)$\wedge \neg$manager(e,d,t-1)

emp(e,d,t)$\leftarrow$ add-emp(e,d,t1) $\wedge$ t1$\leq$t $\wedge \neg \exists$t2 del-emp(e,d,t2) $\wedge$ t2>t1 $\wedge$ t2$\leq$t

Previous deductive rules have to be allowed and in normal form [LT84]. This may require a transformation of the rules using the procedure described in [LT84]. This is also applied to the rules that we describe in following paragraphs.

For each derived predicate p(x) we define a corresponding derived predicate p(x,t) of the deductive conceptual model. In our example, we will have that works-for(e,m,t) is a derived predicate.

For each deductive rule p(x)$\leftarrow$ $L_1(x_1) \wedge ... \wedge L_n(x_n)$ we define a corresponding deductive rule p(x,t)$\leftarrow$ $L_1(x_1,t) \wedge ... \wedge L_n(x_n,t)$ of the deductive conceptual model. In our example we will have the rule: works-for(e,m,t)$\leftarrow$ emp(e,d,t) $\wedge$ manager(m,d,t).

For each integrity constraint ic(x) we define a corresponding integrity constraint ic(x,t) of the deductive conceptual model. The time term t indicates the time of occurrence of the integrity constraint violation. In our example, we will have the integrity constraint ic1(t).

For each integrity rule ic(x)$\leftarrow$ $L_1(x_1) \wedge ... \wedge L_n(x_n)$ we define a corresponding integrity rule ic(x,t)$\leftarrow$ $L_1(x_1,t) \wedge ... \wedge L_n(x_n,t)$ of the deductive conceptual model. In

our example we will have the integrity rule: $ic1(t) \leftarrow emp(e,d,t) \wedge \neg dept(d,t)$.

In operational conceptual models a transition of the IB is induced by the occurrence of a single operation and more than one operation can not occur simultaneously. In a deductive conceptual model transitions of the IB are induced by the passing of time and several external events or operation instances may occur at the same time. Therefore, in the deductive conceptual model, it will be necessary to forbid explicitly simultaneous operation occurrences by means of additional integrity rules.

For each operation Op(x) we define an integrity constraint ic(t) with the following integrity rule: $ic(t) \leftarrow op(x,t) \wedge op(y,t) \wedge x \neq y$. This integrity constraint will forbid two different instances of operation Op to occur at the same time. In our example, one of these integrity constraints will be the following: $ic2(t) \leftarrow$ new-dept(d1,t) $\wedge$ new-dept(d2,t) $\wedge$ d1≠d2.

For each pair of operations $Op_i(x_i)$, $Op_j(x_j)$ we define an integrity constraint ic(t) with the following integrity rule: $ic(t) \leftarrow op_i(x_i,t) \wedge op_j(x_j,t)$. This integrity constraint will forbid two instances of operations Op_i and Op_j to occur at the same time. In our example, one of these integrity constraints will be the following: $ic3(t) \leftarrow$ new-dept(d,t) $\wedge$ hire(e,d1,t).

Besides transforming the conceptual model that describes the planning problem world, we also have to transform accordingly the initial state, the goal to achieve and the solution plans.

The initial state has to be described as a set of base facts of the deductive conceptual model that represent the operations that have occurred before the planning problem is stated. In the case of an empty initial state the corresponding set of base facts will be empty.

A goal described by a formula F will be defined by a new derived predicate G(t) with a deductive rule: $G(t) \leftarrow F'$ where F' stands for the formula F with the addition of a time term t to each predicate of the formula. For example, the goal ∃e,d emp(e,d), will be transformed into: $G(t) \leftarrow \exists e,d\ emp(e,d,t)$. The description of the goal is completed with a final time T_f in which it has to be achieved.

The plan will be obtained in terms of base facts which represent the operations that must occur. The time term of the base facts will give the order of execution of the events. For example, if we have the empty initial state and the goal $G(t) \leftarrow \exists e,d$ emp(e,d,t) which has to be achieved at a time 2, one of the plans we will obtain will be the following: P={new-dept(Marketing,1), hire(John,Marketing,2)}

6 Conclusions and Further Work

In this paper we have presented an approach to the validation of conceptual models. We have defined a set of desirable properties that a conceptual model should satisfy (satisfiability of a conceptual model, liveliness of a predicate, redundancy of integrity constraints, applicability and executability of an operation) and we have shown how these properties can be checked by using plan generation techniques.

The use of planning allows us to check the whole set of properties in a uniform way. The generality of our approach allows us to be independent of the concrete planning method used to check the properties. This facilitates the application of

improvements in the planning research area to our approach.

We have illustrated how to apply our approach by using two concrete planning methods, one proposed for operational conceptual models and the other for deductive ones. In this sense, we have also shown how a method for planning in deductive conceptual models can be used for planning in operational ones.

As further work, we would like to extend our approach by providing explanations when a certain property is not satisfied. In this direction, we believe that by refining the definition of the properties the designer could also be informed about which part of the conceptual model should be changed to satisfy the property. We would also like to define similar properties in a framework where the operations are automatically generated (see e.g. [PO95]) and to study whether they can be validated by planning.

Another extension of our work would be to adapt it to the validation of object-oriented specifications. This can be done by extending current planning methods to the use of object-oriented representations.

We also plan to develop a tool for validating conceptual models following this approach and to integrate our approach into a conceptual model design methodology.

Acknowledgements

We would like to thank Antoni Olivé who encouraged pursuance of this work. We are also grateful to Hendrik Decker, the members of the IS group and the anonymous referees for their helpful comments. This work has been partially supported by PRONTIC CICYT program projects TIC94-0512 and TIC95-0735.

References

[ABC82] Adrion, W.R.; Branstad, M.A.; Cherniavsky, J.C. "Validation, Verification and Testing of Computer Software", ACM Computing Surveys, Vol. 14, No. 2, 159-192, 1982.

[BDM88] Bry, F.; Decker, H.; Manthey, R. "A Uniform Approach to Constraint Satisfaction and Constraint Satisfiability in Deductive Databases", in J. Schmidt et al (eds): *Proc. 1st EDBT*, 488-505, Springer LNCS 303, 1988.

[Bub86] Bubenko, J.A. "Information system methodologies - A research view". In Olle,T.W.; Sol,H.G.; Verrijn-Stuart,A.A. (Eds.) *Information Systems Design Methodologies: Improving the Practice*, 289-318, North-Holland, 1986.

[Che76] Chen, P.P. "The Enitiy-Relational Model. Towards a Unified View of Data", *ACM TODS*, Vol. 1, No. 1, 9-36, 1976.

[CO92] Costal, D.; Olivé, A. "A Method for Reasoning about Deductive Conceptual Models of Information Systems", *Proc. of the CAiSE-92 Conference*, 612-631, Manchester, 1992.

[Cos95] Costal, D. *Un mètode de planificació basat en l'actualització de vistes en bases de dades deductives*, PhD Thesis, Universitat Politècnica de Catalunya, Barcelona, 1995.

[Dal92] Dalianis, H. "A Method for Validating a Conceptual Model by Natural Language Discourse and Generation", *Proc. of the CAiSE-92 Conference*, 425-444, Manchester, 1992.

[DMMP91] Decker, M.; Moerkotte, G.; Müller, H.; Possega,J. "Consistency Driven Planning", *Proc. of the 5th Portuguese Conference on Artificial Intelligence*, 195-209, Albufeira, Portugal, 1991.

[DTU96] Decker, H.; Teniente, E.; Urpí, T. "How to Tackle Schema Validation by View Updating", *To appear in proc. of the EDBT'96*, Avignon, France, 1996.

[FW95] Feenstra R.; Wieringa R. "Validating Database Constraints and Updates Using Automated Reasoning Techniques", in B. Thalheim (ed): *Proc. of the workshop on Semantics in Databases,* 24-32, TR Univ of Cottbus, 1995.

[GSUW94] Gupta, A.; Sagiv, Y.; Ullman, J.D.; Widom, J. "Constraint Checking with Partial Information", *Proc. 13th PoDS*, 45-55, ACM Press, 1994.

[GW93] Gulla, J.A.; Willumsen, G. "Using Explanations to Improve the Validation of Executable Models", *Proc. of the CAiSE-93 Conference*, 118-142, Paris, 1993.

[HM81] Hammer, M.; McLeod, D. "Database Descritpion with SDM: a Semantic Database Model", *ACM TODS*, Vol. 6, No. 3, 351-386, 1981.

[JC92] Jesus, L.; Carapuça, R. "Automatic Generation of Documentation for Information Systems", *Proc. of the CAiSE-92 Conference*, 48-64, Manchester, 1992.

[Kun84] Kung, C.H. *A Temporal Framework for Information Systems Specifications and Verification*, PhD Thesis, Univ. of Trondheim, Norway, 1984.

[Kun85] Kung, C.H. "A Tableaux Approach for Consistency Checking", In Sernadas, A.; Bubenko, J.; Olivé, A. (Eds.) *Information Systems: Theoretical and Formal Aspects*. Elsevier Science Publishers, North-Holland, 191-207, 1985.

[LMSS93] Levy, A.; Mumick, I.S.; Sagiv, Y.; Shmueli, O. "Equivalence, Query-reachability, and Satisfiability in Datalog Extensions", *Proc. 12th PoDS*, 1993.

[LK93] Lindland, O.I.; Krogstie, J. "Validating Conceptual Models by Transformational Prototyping", *Proc. of the CAiSE-93 Conference*, 165-183, Paris, 1993.

[LL93] Lalioti, V,; Loucopoulos, P. "Visualisation for Validation", *Proc. of the CAiSE-93 Conference*, 143-164, Paris, 1993.

[LT84] Lloyd, J.W.; Topor,R.W. "Making Prolog more expressive", *Journal of Logic Programming*, No.3, 225-240, 1984.

[LTP91] Loucopoulos, P.; Theodoulidis, B.; Pantazis, D. "Business rules modelling: conceptual modelling and object-oriented specifications", In Van Assche, F.; Moulin,B.; Rolland,C. (Eds.) *Object Oriented Approach in Information Systems*, North-Holland, 323-342, 1991.

[Lun82] Lundberg, B. "On Correctness of Information Models". *Information Systems*, Vol. 8, No. 2, 87-93, 1983.

[MBW80] Mylopoulos, J.; Bernstein, P.A.; Wong, H.K.T. "A Language Facility for Designing Database-Intensive Applications", *ACM TODS*, Vol. 5, No. 2, 185-207, 1980.

[NH89] Nijssen, G.M,; Halpin, T.A. *Conceptual Schema and Relational Database Design. A Fact Oriented Approach*, Prentice-Hall, 1989.

[Nic82] Nicolas, J.M. "Logic for improving integrity checking in relational databases", *Acta Informatica*, 18, 227-253, 1982.

[OS95] Olivé,A.; Sancho,M.R. "A Method for Explaining the Behaviour of Conceptual Models", *Proc. of the CAiSE-95 Conference*, 12-25, Jyväskylä, 1995.

[PO95] Pastor, J.A.; Olivé,A. "Supporting Transaction Design in Conceptual Modelling of Information Systems", *Proc. of the CAiSE-95 Conference*, 40-53, Jyväskylä, 1995.

[RP92] Rolland, C.; Proix, C. "A Natural Language Approach for Requirements Engineering", *Proc. of the CAiSE-92 Conference*, 257-277, Manchester, 1992.

[VF85] 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*, 175-189, 1985.