Deriving Transaction Specifications from Deductive Conceptual Models of Information Systems

María Ribera Sancho
Antoni Olivé

Report LSI–94–15–R

Deriving Transaction Specifications from Deductive Conceptual Models of Information Systems

(extended version)[1]

Maria Ribera Sancho
Antoni Olivé

Facultat d'Informàtica, Universitat Politècnica de Catalunya
Pau Gargallo 5, 08028 Barcelona - Catalonia
e-mail:{riberalolivé}@lsi.upc.es

Abstract. We review the main components of a Deductive Conceptual Model (DCM) of an IS, and introduce a logic-based language for its specification.

We then present a new, formal method for the derivation of transaction specifications implied by a given DCM. The method is based on the SLDNF proof procedure, and can be implemented easily in Prolog environments.

The method requires the development of an Internal Events Model (IEM). We present such a model, point out how can it be automatically obtained and discuss its use in transaction derivation.

1 Introduction

A general trend in information systems engineering is the adoption of knowledge engineering techniques to enhance existing methodologies as well as to introduce new and more productive development paradigms, methods and supporting tools [Bub86]. This trend is based on the fact that information systems development is a knowledge intensive task [MBJ90] and, thus, it is not surprising that a lot of research has been directed toward providing knowledge-based tools to support the entire information systems development life cycle, from high-level design to low-level code generation [Fre85].

This paper describes our work on deriving transaction specifications from a Deductive Conceptual Model. Our approach uses a logic-based language for the specification of conceptual models, and applies logic-based techniques for the generation of a system design from a conceptual model.

The main originality of this work is that our language is based on the "deductive" approach to conceptual modelling, instead of the traditional, "operational" approach. Both approaches can provide a complete specification of the static and dynamic aspects of an information system, but they differ in the way the dynamic aspect is modelled.

In the operational approach, changes to the Information Base (IB), corresponding to changes in the Universe of Discourse (UoD), are defined by means of operations. The occurrence of a real-world, external event triggers the execution of an operation (transaction), which reflects the effect of the event on the IB. These effects consist

[1] This is an extended version of the paper on "Deriving Transaction Specifications from Deductive Conceptual Models of Information Systems" wich has been accepted for presentation at CAiSE.94 Conference.

usually of insertions, updates or deletions to the IB. On the other hand, operations, as well as queries and integrity constraints, usually have only access to the current state of the IB.

In the deductive approach, the IB is defined only in terms of the external events, by means of deductive rules, and queries and integrity constraints are defined as if the complete history of the IB were available. A deductive conceptual model (DCM) is a specification of an IS in the deductive approach. Examples of conceptual modelling languages using the deductive approach are DADES [Oli82] and CIAM [GKB82].

A detailed comparison of the operational and deductive approaches can be found in [BuO86, Oli86]. The main conclusions are that DCMs provide more local definitions, are easier to change and to accommodate new requirements, and provide more design freedom. However, DCMs are much more difficult to implement than operational models. The reason is that an operational model already embeds some architectural design decisions, which are not made in DCMs [Oli89].

The main design decisions required to implement an information system from a DCM are data base design and transaction design. Usually, many valid alternatives in data base design exist, and the designer must choose the alternative that he/she considers most appropriate. Complete automation of this decision is not possible, but there is a place for CASE tools that aid the designer by confirming the consistency of his/her decisions and by evaluating their impact in terms of performance.

In transaction design, the designer must decide which transactions will exist and, for each of them, when will be executed, its pre-conditions and the actions to be performed, including data base updates and output production.

Contrary to data base design, transaction design from a DCM can be completely automated. We can build a transaction for each external event, to be executed when that event occurs in the real world. Transaction pre-conditions can be determined from the integrity constraints defined in the DCM. Data base updates can be determined from an analysis of deductive rules, and output production can be determined from queries definition.

In this paper we present a new, formal method for generating transaction specifications from a DCM. We extend here the work in the ODISSEA project reported in [San92], where more details on the general framework can be found. The method is based on the use of the SLDNF proof procedure and, thus, it can be implemented easily in Prolog environments.

To our knowledge, there is no similar work to ours in the deductive approach, although there exists some similar research in the context of the operational approach. We can mention here the recent work from the DAIDA project, reported in [CKM91], where additional references to previous research can be found. DAIDA proposes a dependency-based framework for the mapping from a requirements specification into a system design. The framework is dependency-based in the sense that the mapping of parts of the requirements specification is guided by predefined allowable dependencies. At the same time, the framework is goal-oriented, in the sense that non-functional requirements are treated as possibly conflicting goals to be satisfied by the generated design.

In DAIDA, requirements specification prescribe not only the behaviour of the system, but also the environment within which it will function. Instead of this, we focus only on the system to be developed, and our specifications are executable. This allows us to automate part of the mapping from specification to design. On the other hand, we have not considered yet the role of non-functional requirements in the generation of designs.

The paper is organised as follows. Section 2 briefly introduces the main components of a DCM, including an example that is used throughout the paper. Section 3 presents the Internal Events Model (IEM), a key concept in our approach to design generation. Section 4 discusses the use of the IEM in transaction generation, and gives a formal method for deriving transaction specifications from a DCM. Finally, Section 5 summarises the results of our work and points out future research.

2 Deductive Conceptual Models

We characterise the deductive conceptual modelling approach in a first order logic framework.

Time plays a major role in this approach. Every possible information i is associated with a time point T(i), which states when the information holds. We will assume that 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 deductive conceptual model (DCM) of an IS consists of five sets: A set B of base predicates, a set D of derived predicates, a set IC of integrity constraints, a set Q of predefined queries and a set A of alerts. In the following, we briefly describe each of these sets. Figure 1 shows an example that will be used throughout the paper.

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

In the example of figure 1 we have five base predicates: new_person, new_subject, offer, enrol and cancel. A base fact new_person(p,t) means that at time t, p becomes a person. A base fact new_subject(s,t) reports that at time t, s becomes a new subject. A base fact offer(c,ti,s,t) means that at time t, course c is offered with title ti and subject s. A base fact enrol(p,c,t) means that person p is enrolled to course c at time t. Finally, a base fact cancel(c,t) reports that course c is cancelled at time t. We take as time unit a second.

Derived predicates model the relevant types of knowledge about the Universe of Discourse. Each fact of a derived predicate, called derived fact, represents an information about the state of the UoD, at a particular time point. We also assume that the last term gives the time when the information holds. Thus, for example, a derived fact person(p,t) might mean that p is a person at time t.

Each derived predicate is defined by means of one or more deduction rules. A deduction rule of predicate p has the form $p(X_1,..., X_n,T) \leftarrow L_1,..., L_m$, where $p(X_1,..., X_n,T)$ is an atom denoting the conclusion and $L_1,..., L_m$ are literals representing conditions. Each L_j is either an atom or a negated atom. Variables in the conclusion or in the conditions are assumed to be universally quantified over the whole formula. The terms in the conclusion must be distinct variables, and the terms in the conditions must be variables or constants.

Condition predicates may be ordinary or evaluable ("built-in"). The former are base or derived predicates, while the latter are predicates, such as the comparison or arithmetic predicates, that can be evaluated without accessing a database.

We assume every rule to be allowed [GMN84], i.e. any variable that occurs in the rule has an occurrence in positive condition of an ordinary predicate. We also require every rule to be time-restricted. This means that for every positive literal q(...,T1) of a base or derived predicate q occurring in the body, the condition $L_1,..., L_m \rightarrow T1 \leq T$ must hold. This ensures that $p(X_1,..., X_n,T)$ is defined in terms of q-facts holding at time T or before.

In the example, there are six derived predicates, with their corresponding (and hopefully self-explanatory) rules. For the sake of clarity, our examples do not follow strictly the above format, but can be transformed into it using the procedure given in [LIT84].

base predicates
new_person(Person,Time)
new_subject(Subject,Time)
offer(Course,Title,Subject,Time)
enrol(Person,Course,Time)
cancel(Course,Time)
derived predicates
subject(S,T) ← new_subject(S,T1), T1 ≤ T
person(P,T) ← new_person(P,T1), T1 ≤ T
course(C,Ti,T) ← offer(C,Ti,S,T1),T1 ≤ T,
not ∃T2(cancel(C,T2),T2>T1, T2≤ T)
subject_of_course(S,C,T) ← subject(S,T), course(C,Ti,T),
offer(C,Ti,S,T1), T1 ≤ T
enrolled(P,C,T) ← course(C,Ti,T), enrol(P,C,T1), T1 ≤ T
interested(P,S,T) ← enrol(P,C,T1), T1 ≤ T, subject_of_course(S ,C,T1)
integrity constraints
ic1(T) ← new_subject(S,T), subject(S,T-1)
ic2(T) ← new_person(P,T), person(P,T-1)
ic3(C,T) ← offer(C,Ti,S,T), course(C,Ti1,T1),T1<T
ic4(C,S,T) ← offer(C,Ti,S,T), not subject(S,T-1)
ic5(C,T) ← offer(C,Ti,S,T), offer(C,Ti1,S1,T), Ti ≠Ti1
ic6(C,T) ← offer(C,Ti,S,T), offer(C,Ti1,S1,T), Sl ≠S1
ic7(T) ← cancel(C,T), not ∃Ti(course(C,Ti,T-1))
ic8(T) ← enrol(P,C,T), not person(P,T-1)
ic9(T) ← enrol(P,C,T), not ∃Ti(course(C,Ti,T-1))
ic10(T) ← enrol(P,C,T), enrolled(P,C,T-1)

Figure 1. Example of Deductive Conceptual Model

Integrity constraints are closed first-order formulas that base and/or derived facts are required to satisfy. We deal with constraints that have the form of a denial $\leftarrow L_1,...,L_m$, with $m \geq 1$, where the L_j are literals, and variables are assumed to be universally

quantified over the whole formula. More general constraints can be transformed into this form as described in [LIT84]. For the sake of uniformity, we associate to each integrity constraint an inconsistency predicate icn, with at least a time term, and thus it has the same form as the deductive rules. We call them integrity rules.

In the example of figure 1 we show ten inconsistency predicates, with their rules. To see how an inconsistency may arise, assume that base facts new_person(john,10), new_subject(maths,10), offer(c1,ti,maths,15) and enrol(john,c1,18) were received at times 10, 15 and 18, respectively. Thus, the facts that hold at time 19 are: person(john,19), subject(maths,19), course(c1,ti,19), enrolled(john,c1,19) and interested(john,maths,19). Now, if at time 20, the IS receives offer(c1,ti,electronics,20) the inconsistency facts ic3(c1,20) and ic4(c1,electronics,20) will hold, because course(c1,ti,19) does hold and subject(electronics,19) does not hold, respectively. Therefore, the base fact offer(c1,ti,electronics,20) would be rejected.

Outputs from an IS may be requested by the users (queries) or triggered internally by the system when some condition holds (alerts). Each query is defined by a name, a number of parameters, to which the user will give values when he/she makes the query, and a body. The answer to a query is the set of values that satisfies the conditions given in the body. (In order to focus on our objective, we omit in this paper output definition and handling).

As can be observed, there is a strong similarity in form between a DCM and a deductive database. However, there are some fundamental differences between both. An explanation can be found in [Oli89]

3 The Internal Events Model

We have seen, in the previous Section, the main components of a DCM. Now, we start to describe our approach to the design and implementation of an IS from its DCM. The key concept of our approach is the Internal Events Model (IEM)[Oli89,San90]. In the following, we briefly describe the main concepts of an IEM, and show its application to the example.

3.1 Classification of predicates

Predicates defined in a DCM can be classified according to their temporal behaviour. For our purposes, the most important classification is the following. Let p be a predicate and **k** a vector of constants. Assume that fact p(**k**) holds at time T-1. What can we say about the truth of p(**k**) at time T? Three cases are possible:

a) p(**k**) will be true at time T. Then we classify p as P-*steady*.

b) p(**k**) will be false at time T. Then we classify p as P-*momentary*.

c) p(**k**) can be true or false at time T. In this case, assume that no external events happen at time T. We have tree subcases:

c1) p(**k**) will be true at time T. Then we classify p as P-*state*.

c2) p(**k**) will be false at time T. Then we classify p as P-*transient*.

c3) p(**k**) can be true or false at time T, depending on the truth value of some condition that must be evaluated at time T. Then we classify p as P-*spontaneous*.

Base predicates are assumed to be P-transient. In our DCM example of figure 1, predicates "subject", "person" and "interested" are P-steady. The other derived predicates are P-state.

3.2 Internal events

The concept of internal event tries to capture, in a natural way, the notion of change in the extension of a predicate. We associate to each predicate p an insertion internal event ιp, and to each P-state or P-spontaneous derived predicate q a deletion internal event δq.

Insertion internal events are defined as follows. Let p be a P-steady, P-state or P-spontaneous predicate, then:

(1) $\forall \mathbf{X},T\ (\iota p(\mathbf{X},T) \leftrightarrow p(\mathbf{X},T) \wedge \neg p(\mathbf{X},T\text{-}1))$

If p is P-momentary or P-transient, then:

(2) $\forall \mathbf{X},T\ (\iota p(\mathbf{X},T) \leftrightarrow p(\mathbf{X},T))$

Observe that if p is a base predicate, ιp facts represent external events (given by the environment) corresponding to insertions of base facts. If p is a derived predicate, then ιp facts represent induced insertions of derived facts. Finally, if p is an inconsistency predicate, ιp facts correspond to violations of its integrity constraint. In this case, since we assume that the IB is consistent at time T-1, p(X,T-1) is always false, and the literal ¬p(X,T-1) can be removed from (1).

We similarly define *deletion internal events*. Let p be a P-state or P-spontaneous derived predicate, then:

(3) $\forall \mathbf{X},T\ (\delta p(\mathbf{X},T) \leftrightarrow p(\mathbf{X},T\text{-}1) \wedge \neg p(\mathbf{X},T))$

If p is a derived predicate, then δp facts represent induced deletions of derived facts. Note that, for inconsistency predicates, δp facts cannot happen in any transition, since p(X,T-1) is always false.

Rules (1), (2) and (3) are called *internal events rules*. They can be obtained using a transformation of DCM rules, as we will explain in the following.

3.3 Transition rules

We first transform deduction rules of the DCM into a set of equivalent ones, called transition rules. Let p(X,T) be a derived or inconsistency predicate. The definition of p consists of the rules in the DCM having p in its conclusion. Assuming that there are $m \geq 1$ such rules, we rename the conclusions of the m rules by $p_1,...,p_m$, change the implication by an equivalency, and add the set of clauses:

$p(\mathbf{X},T) \leftarrow p_i(\mathbf{X},T)$ for i = 1..m

In the example, as predicate "enrolled" is defined with one rule, we would write:

$enrolled_1(P,C,T) \leftrightarrow course(C,Ti,T), enrol(P,C,T1), T1 \leq T$

$enrolled(P,C,T) \leftarrow enrolled_1(P,C,T)$

Consider now one of the rules $p_i(\mathbf{X},T) \leftrightarrow W$, being W a set of literals $L_1,...,L_n$. The idea consists on replacing each literal L_r of W, corresponding to a base or derived predicate, and whose time variable may range in the rule over a set including T, by an equivalent expression containing internal events predicates, whose time variables range only over {T}, and base or derived predicates, whose time variables range over a set not including T. The rule obtained by this transformation is a *transition rule*.

The transformation applied to each ordinary literal L_r of W is based on the rules (1), (2) and (3). It depends on the P-type of its predicate, its sign (positive or negative) and the range of its time variable. We say that an ordinary literal L_r is

a) *Current*, if its time variable ranges only over {T}.

b) *Past*, if its time variable ranges over a set including T.

c) *Old*, if its time variable ranges over a set not including T.

For example, in the rule:

$enrolled_1(P,C,T) \leftrightarrow course(C,Ti,T), enrol(P,C,T1), T1 \leq T$

literal enrol(P,C,T1) is past because T1 may range over the set $\{T_0,...,T\}$, where T_0 is the initial time, and literal course(C,Ti,T) is current because its time variable ranges only over {T}.

We will explain in detail the transformation applied to current and past literals corresponding to P-state or P-spontaneous predicates. The rules corresponding to the other P-types can be similarly deduced. Notice that old literals do not have to be transformed.

Let L_r=q(**X**,T) be a current literal of W. From (1) and (3) we can directly obtain a set of equivalencies that define the value of q(**x**) (respectively ¬q(**x**)) at time t in terms of its value at time t-1 and the internal events happened at t, as follows.

If q is P-state or P-spontaneous:

(4) $\forall \mathbf{X},T \ (q(\mathbf{X},T) \leftrightarrow (q(\mathbf{X},T\text{-}1) \wedge \neg \delta q(\mathbf{X},T)) \vee \iota q(\mathbf{X},T))$

$\forall \mathbf{X},T \ (\neg q(\mathbf{X},T) \leftrightarrow (\neg q(\mathbf{X},T\text{-}1) \wedge \neg \iota q(\mathbf{X},T)) \vee \delta q(\mathbf{X},T))$

They mean that q(**x**) is true at time t if it was true at time t-1 and has not been deleted at t, or if it has been inserted at t; and that q(**x**) is false at t if it was false at t-1 and has not been inserted at t, or if it has been deleted at t. With these rules we already can transform current literals.

For the case of past literals L_r=q(**X**,T1) (with T1≤T) we just have to consider separately the case in which T1<T and the case in which T1=T. Then:

$\forall \mathbf{X},T \ (q(\mathbf{X},T1) \leftrightarrow (T1<T \wedge q(\mathbf{X},T1)) \vee (T1=T \wedge q(\mathbf{X},T)))$

$\forall \mathbf{X},T \ (\neg q(\mathbf{X},T1) \leftrightarrow (T1<T \wedge \neg q(\mathbf{X},T1)) \vee (T1=T \wedge \neg q(\mathbf{X},T)))$

and replacing q(**X**,T) and ¬q(**X**,T) by their equivalent definition given in (4). We obtain:

$\forall \mathbf{X},T \ (q(\mathbf{X},T1) \leftrightarrow ((T1<T \wedge q(\mathbf{X},T1)) \vee$
$(T1=T \wedge q(\mathbf{X},T\text{-}1) \wedge \neg\delta q(\mathbf{X},T)) \vee$
$(T1=T \wedge \iota q(\mathbf{X},T)))$

$\forall \mathbf{X},T \ (\neg q(\mathbf{X},T1) \leftrightarrow ((T1<T \wedge \neg q(\mathbf{X},T1)) \vee$
$(T1=T \wedge \neg q(\mathbf{X},T\text{-}1) \wedge \neg \iota q(\mathbf{X},T)) \vee$
$(T1=T \wedge \delta q(\mathbf{X},T)))$

In Figure 2 we show the transformation to be applied in each case. The words O, Pr and N are mnemotechnic of Old, Previous and New. $O(L_r)$, $Pr(L_r)$ and $N(L_r)$ represent the conditions upon which a literal L_r of W is true at time T. $O(L_r)$, if not false, corresponds to the case in which L_r is true at T because it was true some time in the past. $Pr(L_r)$, if not false, covers the case in which L_r is true at T because it was true at T-1. Finally $N(L_r)$, if not false, represents the case in which L_r is true at T because it has been induced by the base facts happened at T, and was false at T-1.

Replacing each each literal L_r of P_i by its equivalent expression we obtain:

$$P_i(\mathbf{X},T) \leftrightarrow \bigwedge_{r=1}^{r=n} [O(L_r) \vee Pr(L_r) \vee N(L_r) \mid L_r] \qquad \text{for } i = 1..m$$

where the first option is taken if L_r is a current or past ordinary literal, and the second one if it is old or evaluable.

		P-type of q	O (L_r)	Pr (L_r)	N (L_r)
L_r positive	L_r current	state or spontaneous	false	$q(X,T-1) \wedge \neg \delta q(X,T)$	$\iota q(X,T)$
		steady	false	$q(X,T-1)$	$\iota q(X,T)$
		transient or momentary	false	false	$\iota q(X,T)$
	L_r past	state or spontaneous	$T1<T \wedge q(X,T1)$	$T1=T \wedge q(X,T-1) \wedge \neg \delta q(X,T)$	$T1=T \wedge \iota q(X,T)$
		steady	$T1<T \wedge q(X,T1)$	$T1=T \wedge q(X,T-1)$	$T1=T \wedge \iota q(X,T)$
		transient or momentary	$T1<T \wedge q(X,T1)$	false	$T1=T \wedge \iota q(X,T)$
L_r negative	L_r current	state or spontaneous	false	$\neg q(X,T-1) \wedge \neg \iota q(X,T)$	$\delta q(X,T)$
		steady	false	$\neg q(X,T-1) \wedge \neg \iota q(X,T)$	false
		transient or momentary	false	false	$\neg \iota q(X,T)$
	L_r past	state or spontaneous	$T1<T \wedge \neg q(X,T1)$	$T1=T \wedge \neg q(X,T-1) \wedge \neg \iota q(X,T)$	$T1=T \wedge \delta q(X,T)$
		steady	$T1<T \wedge \neg q(X,T1)$	$T1=T \wedge \neg q(X,T-1) \wedge \neg \iota q(X,T)$	false
		transient or momentary	$T1<T \wedge \neg q(X,T1)$	false	$T1=T \wedge \neg \iota q(X,T)$

Figure 2. Transformation of ordinary literals

Distributing $\wedge$ over $\vee$, we get an equivalent set of transition rules with the general form:

$$P_{i,k}(\mathbf{X},T) \leftrightarrow \bigwedge_{r=1}^{r=n} [O(L_r) \mid Pr(L_r) \mid N(L_r)] \wedge E(T) \qquad \text{for } k = 1 \ldots 3^j$$

where j is the number of current or past ordinary literals appearing in the rule $p_i(X,T)$, and E(T) is the conjunction of all old and evaluable literals of $p_i(X,T)$. Then the set of transition rules for predicate P_i are:

(5) $P_i(\mathbf{X},T) \leftarrow P_{i,k}(\mathbf{X},T)$ for $k = 1 \ldots 3^j$

If we apply this transformation to predicate "enrolled", after eliminating the rules with a "false" literal, we obtain the following transition rules:

$enrolled_{1,1}(P,C,T) \leftarrow course(C,Ti,T-1), \text{not } \delta course(C,Ti,T), enrol(P,C,T1), T1 < T$
$enrolled_{1,2}(P,C,T) \leftarrow course(C,Ti,T-1), \text{not } \delta course(C,Ti,T), \iota enrol(P,C,T), T1 = T$
$enrolled_{1,3}(P,C,T) \leftarrow \iota course(C,Ti,T), enrol(P,C,T1), T1 < T$
$enrolled_{1,4}(P,C,T) \leftarrow \iota course(C,Ti,T), \iota enrol(P,C,T), T1 = T$

Note that these rules allow us to infer $enrolled_1$ facts holding at time t in base to the course and enrol facts holding at time t-1 or before, and the events ιenrol, ιcourse and δcourse that occur at time t.

3.4 Internal events rules

Once the transition rules have been obtained, we can derive the internal events rules. An internal event rule is a rule that defines the conditions upon which an internal event happens in a given transition. For example, the rules:

ιcourse(C,Ti,T) ← ιoffer(C,Ti,S,T)

δcourse(C,Ti,T) ← course(C,Ti,T-1), ιcancel(C,T)

are an insertion and a deletion internal event rule, respectively. The first states that the occurrence of an ιoffer fact (in this case, the insertion of an offer base fact) induces a corresponding ιcourse fact. The second rule states that the occurrence of a fact ιcancel(c) at time t induces a fact δcourse(c,ti) at time t if course(c,ti) was true at previous time.

We get the internal event rules corresponding to a predicate p using a procedure based on the definitions (1), (2) and (3), as follows.

Assume that predicate p is defined with m rules. Then

$p(\mathbf{X},T) \leftrightarrow p_1(\mathbf{X},T) \vee ... \vee p_m(\mathbf{X},T)$

and rules (1) (2) (3) can be rewritten as, for i = 1..m:

If p is P-steady, P-state or P-spontaneous:

(6) $\forall \mathbf{X},T\ (\iota p(\mathbf{X},T) \leftarrow p_i(\mathbf{X},T) \wedge \neg p_1(\mathbf{X},T\text{-}1) \wedge ... \wedge \neg p_i(\mathbf{X},T\text{-}1) \wedge ... \wedge \neg p_m(\mathbf{X},T\text{-}1))$

If p is P-momentary or P-transient:

(7) $\forall \mathbf{X},T\ (\iota p(\mathbf{X},T) \leftarrow p_i(\mathbf{X},T))$

If p is P-state or P-spontaneous:

(8) $\forall \mathbf{X},T\ (\delta p(\mathbf{X},T) \leftarrow p_i(\mathbf{X},T\text{-}1) \wedge \neg p_1(\mathbf{X},T) \wedge ... \wedge \neg p_i(\mathbf{X},T) \wedge ... \wedge \neg p_m(\mathbf{X},T))$

Rules (6) and (7) are called insertion internal events rules for predicate p. These rules allow us to deduce which facts ιp (induced insertions) occur at any time point t. Similarly, rules (8) are called deletion internal events rules, and represent which facts δp (induced deletions) occur at any time t.

Consider now each p_i separately. Observe that $p_i(\mathbf{X},T) \wedge \neg p_i(\mathbf{X},T\text{-}1)$ corresponds to insertions of p_i. We denote it as $\iota p_i(\mathbf{X},T)$. Similarly, $p_i(\mathbf{X},T\text{-}1) \wedge \neg p_i(\mathbf{X},T)$ corresponds to deletions of p_i, and we denote it as $\delta p_i(\mathbf{X},T)$. Now, substituting $p_i(\mathbf{X},T)$ by its equivalent definition given by the corresponding transition rules (5), we have that (6), (7) and (8) can be rewritten as:

if p is P-steady, P-state or P-spontaneous:

$\forall \mathbf{X},T\ (\iota p(\mathbf{X},T) \leftarrow \iota p_i(\mathbf{X},T) \wedge$
$\neg p_1(\mathbf{X},T\text{-}1) \wedge ... \wedge \neg p_{i-1}(\mathbf{X},T\text{-}1) \wedge \neg p_{i+1}(\mathbf{X},T\text{-}1) \wedge ... \wedge \neg p_m(\mathbf{X},T\text{-}1))$

$\forall \mathbf{X},T\ (\iota p_i(\mathbf{X},T) \leftarrow p_{i,k}(\mathbf{X},T) \wedge \neg p_i(\mathbf{X},T\text{-}1))$

if p is P-momentary or P-transient:

$\forall \mathbf{X},T\ (\iota p(\mathbf{X},T) \leftarrow \iota p_i(\mathbf{X},T))$

$\forall \mathbf{X},T\ (\iota p_i(\mathbf{X},T) \leftarrow p_{i,k}(\mathbf{X},T))$

if p is P-state or P-spontaneous:

$\forall \mathbf{X},T\ (\delta p(\mathbf{X},T) \leftarrow \delta p_i(\mathbf{X},T) \wedge$
$\neg p_1(\mathbf{X},T) \wedge ... \wedge \neg p_{i-1}(\mathbf{X},T) \wedge \neg p_{i+1}(\mathbf{X},T) \wedge ... \wedge \neg p_m(\mathbf{X},T))$

$\forall \mathbf{X},T\ (\delta p_i(\mathbf{X},T) \leftrightarrow p_i(\mathbf{X},T\text{-}1) \wedge \neg p_{i,1}(\mathbf{X},T) \wedge ... \wedge \neg p_{i,k}(\mathbf{X},T))$

for i = 1..m , k = 1... 3^j

Applying the procedure to predicate "enrolled", the insertion internal events rules would be:

$\iota enrolled(P,C,T) \leftarrow \iota enrolled_1(P,C,T)$
$\iota enrolled_1(P,C,T) \leftarrow course(C,Ti,T\text{-}1),\ not\ \delta course(C,Ti,T),\ enrol(P,C,T1),\ T1<T,$
$not\ enrolled_1(P,C,T\text{-}1)$
$\iota enrolled_1(P,C,T) \leftarrow course(C,Ti,T\text{-}1),\ not\ \delta course(C,Ti,T),\ \iota enrol(P,C,T),\ T1=T,$
$not\ enrolled_1(P,C,T\text{-}1)$
$\iota enrolled_1(P,C,T) \leftarrow \iota course(C,Ti,T),\ enrol(P,C,T1),\ T1<T,$
$not\ enrolled_1(P,C,T\text{-}1)$
$\iota enrolled_1(P,C,T) \leftarrow \iota course(C,Ti,T),\ \iota enrol(P,C,T),\ T1 = T,$
$not\ enrolled_1(P,C,T\text{-}1)$

3.5 Simplification of the Internal events rules

The procedure described above obtains an IEM from a given DCM. However, the resulting set of internal events rules can be substantially simplified. Take, for instance the first rule for $\iota enrolled_1$. The rule can not produce any ιenrolled fact, since course(C,Ti,T-1), enrol(P,C,T1), T1<T implies $enrolled_1$(P,C,T-1) and, as a consequence, it can be dropped from the IEM.

IDR.1 ιsubject(S,T) ← ιnew_subject(S,T)
IDR.2 ιperson(P,T) ← ιnew_person(P,T)
IDR.3 ιcourse(C,Ti,T) ← ιoffer(C,Ti,S,T)
IDR.4 δcourse(C,Ti,T) ← course(C,Ti,T-1), ιcancel(C,T)
IDR.5 ιsubject_of_course(S,C,T) ← ιoffer(C,Ti,S,T)
IDR.6 δsubject_of_course(S,C,T) ← subject_of_course(S,C,T-1), δcourse(C,Ti,T)
IDR.7 ιenrolled(P,C,T) ← course(C,Ti,T-1), not δcourse(C,Ti,T), ιenrol(P,C,T)
IDR.8 δenrolled(P,C,T) ← enrolled(P,C,T-1), δcourse(C,Ti,T)
IDR.9 ιinterested(P,S,T) ← ιenrol(P,C,T), subject_of_course(S,C,T-1),
not δsubject_of_course(S,C,T), not interested(P,S,T-1)
IDR.10 ιic1(T) ← ιnew_subject(S,T), subject(S,T-1)
IDR.11 ιic2(T) ← ιnew_person(P,T), person(P,T-1)
IDR.12 ιic3(C,T) ← ιoffer(C,Ti,S,T), course(C,Ti1,T1),T1<T
IDR.13 ιic4(C,S,T) ← ιoffer(C,Ti,S,T), not subject(S,T-1)
IDR.14 ιic5(C,T) ← ιoffer(C,Ti,S,T), ιoffer(C,Ti1,S1,T), Ti ≠Ti1
IDR.15 ιic6(C,T) ← ιoffer(C,Ti,S,T), ιoffer(C,Ti1,S1,T), Sl ≠S1
IDR.16 ιic7(T) ← ιcancel(C,T), not ∃Ti(course(C,Ti,T-1))
IDR.17 ιic8(T) ← ιenrol(P,C,T), not person(P,T-1)
IDR.18 ιic9(T) ← ιenrol(P,C,T), not ∃Ti(course(C,Ti,T-1))
IDR.19 ιic10(T) ← ιenrol(P,C,T), enrolled(P,C,T-1)

Figure 3. Internal events model of the DCM example.

We have developed a method to simplify the IEM [San93]. It can be described as a procedure that uses all the relevant information about the DCM and the IEM to transform each internal event rule into a semantically equivalent one that can be

evaluated much more efficiently. This transformation is mainly based on the use of the integrity constraints, but it also takes into account the P-type of derived predicates, and the proper deduction rules. The explanation of the method is out of the scope of this paper. Figure 3 shows the result obtained by its application to the DCM example of figure 1.

4. Deriving Base Transactions

4.1 Implementation of a DCM

Implementation of a DCM comprises two main design decisions: data base and transactions. The simplest implementation consists in storing in the data base (Extensional Data Base, EDB) only the base facts. There is a transaction (that we call Base Transaction, BT) for each base predicate. The role of each BT is to read the corresponding base fact, check the relevant integrity constraints and store the fact in the EDB. Derived facts could be computed using the DCM rules when requested, either during constraints checking or query answering. This implementation, however, would not be acceptable in practical situations from an efficiency point of view.

In general, the EDB comprises all facts explicitly stored in the data base. These facts do not need to be the base facts defined in the DCM. Indeed, it is quite usual to store in the EDB some derived facts, and not to store all base facts. Usually, there are many valid alternatives and the designer must choose in each case the most appropriate. The EDB schema is characterized by: (1) A set of base or derived predicates of the DCM to be included in the EDB (we call them *stored predicates*), and (2) for each of them, a time interval for which its facts will be stored. The Intensional Data Base, IDB, comprises all facts that can be derived from the EDB using the DCM rules.

Then, the role of each BT is to read the corresponding base fact, check the relevant integrity constraints and update the EDB (inserting and/or deleting one or more EDB-facts). In this section, we present a method for deriving the specifications of each BT. These specifications include two parts:

(1) The transaction pre-conditions, which is the set of conditions the base fact must satisfy in order to guarantee EDB integrity.

(2) The transaction updates, which are the insertions and/or deletions that must be done to the EDB.

The design and implementation of each BT will then be dependent on the particular execution environment. We will not deal with this aspect here, but we mention that it is quite straightforward to build such transactions on top of a deductive DBMS (see [San92, MSS92] for more details).

4.2 The approach

As we mentioned before, there is a BT for each base predicate defined in the DCM. We can also derive composite transactions, corresponding to the simultaneous occurrence of two or more base facts, but this extension will not be considered here.

Let TR={ιb(**k**,t)} be the transaction corresponding to the insertion of a base fact b(**k**,t), where **k** is a vector of constants denoting the transaction parameters, and t the occurrence time. We derive its effects in the following way: For each integrity constraint icj specified in the DCM, we check its violation by evaluating the internal event predicate ιicj. In the same way, for each stored predicate p(**X**,T), we derive the

induced insertion (resp. deletion) of a fact p(x,t) by evaluating the internal event predicate $\iota p(\mathbf{X},t)$ (resp. $\delta p(\mathbf{X},t)$). As IEM rules are defined in terms of data base predicates extensions at time t-1 or before, to evaluate internal event predicates we also need the IDB rules, which relate DCM predicates to the facts stored in the EDB.

To obtain these evaluations we use an SLDNF-based proof procedure. More precisely, let $I=[\iota/\delta]p(\mathbf{X},t)$ be the internal event predicate to be evaluated. We say that TR induces I if goal $\{\leftarrow I\}$ succeeds from input set IEM $\cup$ IDB $\cup$ EDB $\cup$ TR. If every branch of the SLDNF-search space for IEM $\cup$ IDB $\cup$ EDB $\cup$ TR $\cup$ $\{\leftarrow I\}$ is a failure branch, then TR does not induce I.

Obviously, the evaluation can be completely done at execution time, when the concrete values of EDB and transaction parameters are known. However, we can do some preparatory work at compilation time, by partially evaluating IEM $\cup$ TR wrt I [LlS91]. The result of the partial evaluation is a set E of n rules ($n \geq 0$):

$I \leftarrow C_i \qquad i = 0..n$

where C_i is a conjunction of literals such that the evaluation of $\{\leftarrow I\}$ at execution time from E $\cup$ IDB $\cup$ EDB gives the same result as the evaluation of $\{\leftarrow I\}$ from IEM $\cup$ IDB $\cup$ EDB $\cup$ TR. If n=0 then TR does not induce I. If C_i is empty then TR unconditionally induces a fact I. In this last case, if I corresponds to the insertion of an inconsistency predicate, the transaction can never happen because it violates the corresponding integrity constraint.

Applying this procedure to our example, at compilation time, assuming that the current state of all derived predicates are stored in the EDB, we have to partially evaluate, for each possible transaction, internal event predicates $\iota ic1(t)$, $\iota ic2(t)$, $\iota ic3(C,t)$, $\iota ic4(C,S,t)$, $\iota ic5(C,t)$, $\iota ic6(C,t)$, $\iota ic7(t)$, $\iota ic8(t)$, $\iota ic9(t)$, $\iota ic10(t)$, corresponding to the integrity constraints, and $\iota subject(S,t)$, $\iota person(P,t)$, $\iota course(C,Ti,t)$, $\delta course(C,Ti,t)$, $\iota subject\_of\_course(S,C,t)$, $\delta subject\_of\_course(S,C,t)$, $\iota enrolled(P,C,t)$, $\delta enrolled(P,C,t)$, and $\iota interestedf(P,S,t)$ to derive the update conditions of stored predicates.

Take, as an example, the transaction: TR={$\iota enrol(p,c,t)$}, where p and c are parameters and t the occurrence time of TR. After application of our method, the set of rules obtained at compilation time will be:

$\iota ic8(t) \leftarrow$ not person(p,t-1)
$\iota ic9(t) \leftarrow$ not $\exists$Ti(course(c,Ti,t-1))
$\iota ic10(t) \leftarrow$ enrolled(p,c,t-1)
$\iota enrolled(p,c,t) \leftarrow$ course(c,Ti,t-1)
$\iota interested(p,S,t) \leftarrow$ subject_of_course(S,c,t-1), not interested(p,S,t-1)

meaning that constraint ic8 will be violated if p was not a person at previous time, constraint ic9 will be violated if c was not a course at previous time and constraint ic10 will be violated if p is already enrolled to course c. If the constraints are satisfied, the fact enrolled(p,c) must be inserted into the EDB if c was a course with title Ti at previous time, and the fact interested(p,S) must be inserted into the EDB if it was not already true, being S the previous value of subject_of_course(S,c). Note, however, that $\iota enrolled(p,c,t)$ can be unconditionally induced, instead of depending on the condition course(c,Ti,t-1), because ic9 guarantees that there exists some course(c,Ti,t-1).

4.3 Example

Before giving the formal definition of the method, we illustrate our approach with an example.

Assume that we want to derive the transaction TR={ιenrol(p,c,t)} and that the current state of all derived predicates is stored in the EDB. We have first to partially evaluate IEM ∪ TR wrt ιicj, corresponding to the integrity constraints. Consider, for example, the partial evaluation wrt ιic8(t). Possible sets of conditions C_i that would lead to the violation of ic8(t) are obtained by having some failed derivation of IEM ∪ TR ∪ {←ιic8(t)} succeed. Figure 4 shows this derivation tree, were the circled labels are references to the rules of the method, defined in section 4.4.

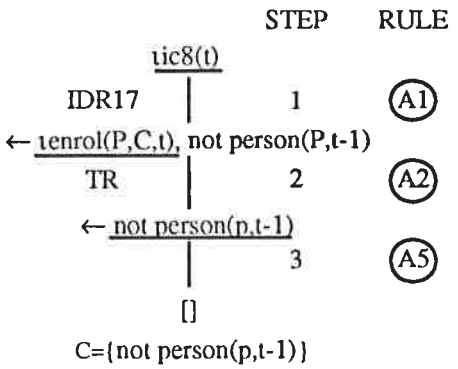

Figure 4: derivation tree for ιic8(t)

Steps 1 and 2 are SLDNF resolution steps. At step 1 rule IDR17 of the IEM acts as input clause, while at step 2 the input clause is the base fact from TR. At step 3 the selected literal not person(p,t-1) corresponds to an stored predicate and, therefore, it can not be evaluated at compilation time because it depends on the concrete value of the database at execution time. As a consequence, this literal is included in the condition set C and its evaluation is delayed until execution time.

The set of conditions obtained for ιic8(t) adds the following rule to the transaction definition:

ιic8(t) ← not person(p,t-1)

The partial evaluation wrt ιic9(t) and ιic10(t) is shown in figure 5.

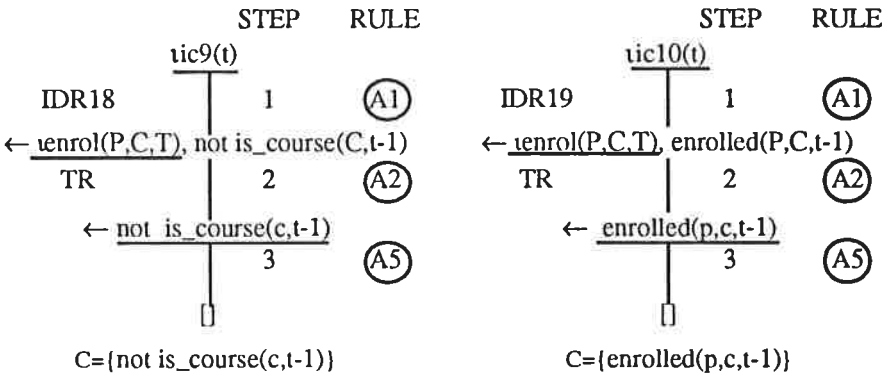

Figure 5: derivation trees for ιic9(t) and ιic10(t)

Figure 7 shows the failure tree for δcourse(c,Ti,t). Steps 1 and 2 are SLDNF resolution steps. At step 1 rule IDR4 of the IEM acts as input clause. At step 2, the selected literal is the external event ιcancel(c,t) which has to be resolved with facts in TR. Given that there is no fact ιcancel(c,t) in TR, the tree fails unconditionally.

As a consequence, the following rule will be added to the BT definition:
ιenrolled(p,c,t) ← course(c,Ti,t-1)

The partial evaluation wrt ιinterested(P,S,t) is shown in figure 8.

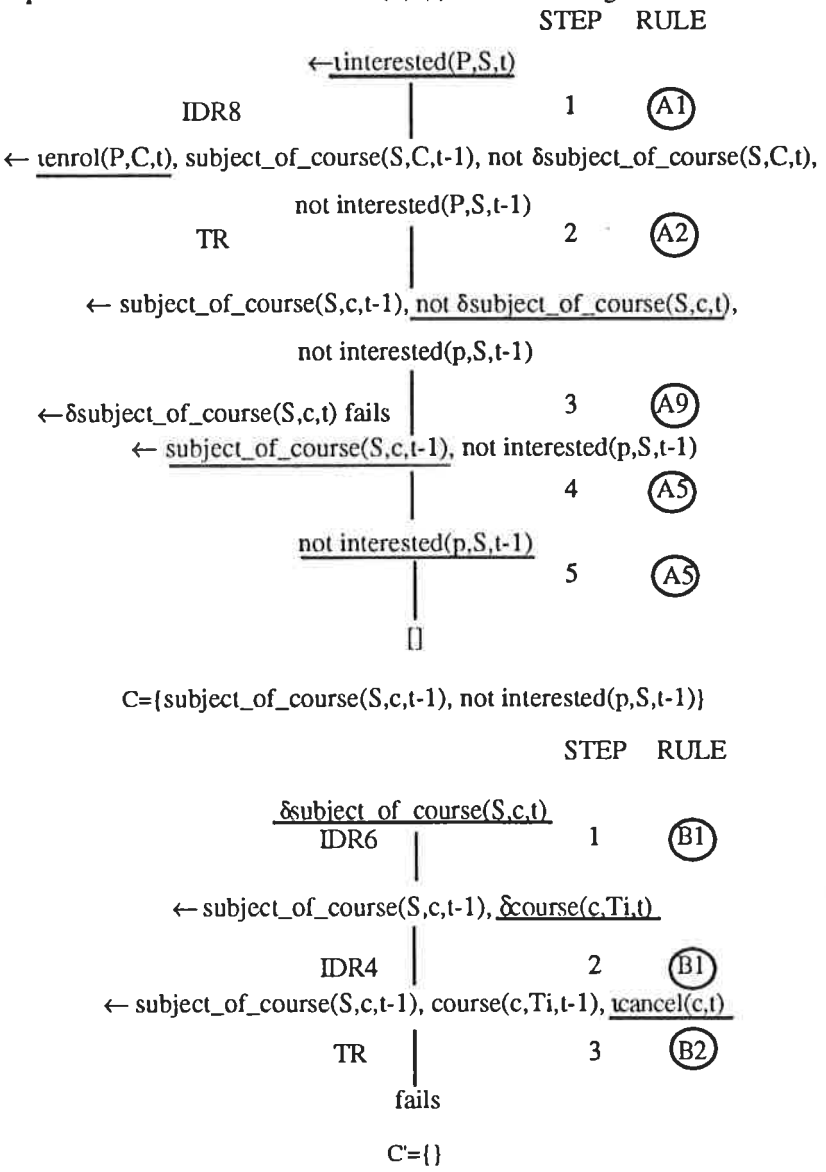

Figure 8: derivation tree for ιinterestec(P,S,t)

4.4 The method

This section describes, in a formal way, the method explained in sections 4.2 and 4.3. Our proposal is based on the partial evaluation procedure given in [LIS91]. The main difference consists on the treatment given to the case in which a negative and non-ground literal is selected during the SLDNF refutation. In some cases, for our particular application, the failure of such kind of literals can be guaranteed by the subsidiary derivation and, therefore, they can be evaluated at compilation time.

As we know, the transaction will be derived by partially evaluating IEM∪TR wrt internal events predicates corresponding to integrity constraints and stored predicates.

If I is the internal event predicate to be evaluated, C will be a set of conditions of I if there exists a **constructive derivation** from (I {}) to ([] C), having as input set IEM∪TR, where TR={ιb(k,t)}, being b a base predicate and **k** a vector of constants.

We will call:

- Non-event literal to a literal corresponding to a base, derived or inconsistency predicate.
- Accessible literal to a non-event literal corresponding to an stored predicate or to a predicate that can be deduced from stored predicates.

Constructive derivation. A constructive derivation from $(I_1\ C_1)$ to $(I_n\ C_n)$ via a selection rule R, that selects literals not corresponding to evaluable predicates with priority, is a sequence:

$(I_1\ C_1)$, $(I_2\ C_2)$,..., $(I_n\ C_n)$

such that for each $i\geq 1$, I_i has the form $\leftarrow L_1,...,L_k$, $R(I_i)=L_j$ and $(I_{i+1}\ C_{i+1})$ is obtained according to one of the following rules:

A1) If L_j is a positive internal event or transition literal, and S is the resolvent of some clause in IEM with I_i on the selected literal L_j, then $I_{i+1}=S$, and $C_{i+1}=C_i$.

A2) If L_j is a positive, external event literal ιp(X,t), and S is the resolvent of fact in TR with I_i on the literal L_j using substitution σ, then $I_{i+1}=S$, and $C_{i+1}=(C_i)\sigma$.

A3) If L_j is a negative, external event literal "not ιp(X,t)", and TR can not be unified with ιp(X,t) then $I_{i+1}= \leftarrow L_1,...,L_{j-1}, L_{j+1},..., L_k$, and $C_{i+1}=C_i$.

A4) If L_j is a negative, external event literal "not ιp(X,t)", and TR can be unified with ιp(X,t) using substitution $\sigma=\{X_1/k_1\}$, then $I_{i+1}= \leftarrow L_1,...,L_{j-1}, L_{j+1},...,L_k$, and $C_{i+1}=C_i \cup \{ X_1\neq k_1\}$.

A5) If L_j is an accessible literal then $I_{i+1}=\leftarrow L_1,...,L_{j-1},L_{j+1},...,L_k$, and $C_{i+1}=C_i\cup\{L_j\}$.

A6) If L_j is an evaluable literal then $I_{i+1}=\leftarrow L_1,...,L_{j-1}, L_{j+1},...,L_k$, and $C_{i+1}=C_i\cup\{L_j\}$.

A7) If L_j is a ground internal event or transition literal "not Q" and there exists a **consistency derivation** from (←Q { }) to ([] C') then

$I_{i+1} = \leftarrow L_1,..., L_{j-1}, L_{j+1},..., L_k$, and $C_{i+1} = C_i\cup C'$.

A8) If L_j is a ground internal event or transition literal "not Q" and it does not exist a **consistency derivation** from (←Q { }) to ([] C') then

$I_{i+1} = \leftarrow L_1,..., L_{j-1}, L_{j+1},..., L_k$, and $C_{i+1} = C_i\cup \{L_j\}$.

A9) If L_j is a non-ground internal event or transition literal "not Q" and there exists a **consistency derivation** from (←Q { }) to ([] { }) then

$I_{i+1} = \leftarrow L_1,..., L_{j-1}, L_{j+1},..., L_k$, and $C_{i+1} = C_i$.

A10) If L_j is a non-ground internal event or transition literal "not Q" and it does not exist a **consistency derivation** from $(\leftarrow Q\ \{\})$ to $([]\ \{\})$ then

$I_{i+1} = \leftarrow L_1,..., L_{j-1}, L_{j+1},..., L_k$, and $C_{i+1} = C_i \cup \{L_j\}$.

The step corresponding to rule A1) is an SLDNF resolution step. The step of rule A2) is also an SLDNF step, but now L_j is resolved with TR. In rule A3) not $\iota p(\mathbf{X},t)$ is true because $\iota p(\mathbf{X},t)$ can not be unified with TR.

In A4) not $\iota p(\mathbf{X},t)$ will be true only if $\mathbf{X_1 \neq k_1}$. Assume, for example, that the current goal is $\leftarrow$ not ιenrol(P,c,t),α and that TR={ιenrol(p,c,t)}. By applying A4) we get the new goal $\leftarrow P \neq p,\alpha$.

At steps A5) and A6) accessible and evaluable literals are added to the condition set C because their evaluation have to be delayed until execution time. Steps corresponding to rules A7) and A8) deal with the evaluation of ground and negative internal event or transition literals "not Q". If the failure of Q can be ensured via a consistency derivation (A7), the set of conditions is added to the condition set C. On the contrary (A8), the literal itself has to be added to the condition set C because it can not be evaluated at compilation time. Finally, steps corresponding to rules A9) and A10) perform the evaluation of non-ground and negative internal event or transition literals "not Q". If a consistency derivation ensures an unconditional failure of Q (A9), the selected literal can not fail, and thus, it can be eliminated from the goal I in order to have a failed derivation of IEM $\cup$ TR $\cup$ {$\leftarrow$I} succeed. On the contrary (A10), the literal has to be added to the condition set C because its evaluation has to be delayed until execution time.

There are different ways in which a constructive derivation can succeed. Each one may lead to different insertion or deletion rule.

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

$(F_1\ C_1), (F_2\ C_2),..., (F_n\ C_n)$

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

B1) If L_j is a positive internal event or transition literal, and S' is the set of all resolvents of clauses in IEM with $\leftarrow L_1,..., L_k$ on the literal L_j, then $F_{i+1} = S' \cup F'_i$ and $C_{i+1}=C_i$.

B2) If L_j is a positive external event literal , and TR can not be unified with L_j then $F_{i+1}=F'_i$, and $C_{i+1}=C_i$.

B3) If L_j is a ground positive external event literal $\iota p(x,t)$, and S' is the resolvent of TR with $\leftarrow L_1,..., L_k$ on the literal L_j, and $[] \notin S'$, then $F_{i+1} = S' \cup F'_i$ and $C_{i+1}=C_i$.

B4) If L_j is a ground negative external event literal "not $\iota p(x,t)$" and TR can be unified with $\iota p(x,t)$ then $F_{i+1} = F'_i$, and $C_{i+1}=C_i$.

B5) If L_j is a negative external event literal "not $\iota p(\mathbf{X},t)$", and TR can not be unified with $\iota p(\mathbf{X},t)$ then $F_{i+1} = \leftarrow L_1,..., L_{j-1}, L_{j+1},..., L_k \cup F'_i$, and $C_{i+1}=C_i$.

B6) If L_j is a ground negative internal event or transition literal "not Q" and there exists a **constructive derivation** from $(\leftarrow Q\ \{\})$ to $([]\ C')$ then $F_{i+1}=F'_i$, and $C_{i+1}= C_i \cup C'$.

Steps corresponding to rules B1) and B3) are SLDNF resolution steps. In case B2) and B4) the current branch fails and thus, it can be eliminated. In case B5) the selected

literal can not fail, and thus, it is eliminated from the subgoal set F_i in order to make a successful SLDNF branch fail. Finally, in case B6) the current branch can be dropped if there exists a constructive derivation for the negation of the selected literal. This ensures failure for it.

5 Conclusions

We have presented the main components of a Deductive Conceptual Model (DCM). We have then discussed the main decisions involved in the design and implementation process: data base design and transaction design. Data base design can not be completely automated, although some tools to aid the designer in alternatives' generation and analysis can be built.

Once the data base has been designed, transactions can be derived automatically. Preconditions of the transactions can be determined from the integrity rules, and database updates from the deductive rules of the DCM. We have presented a formal method to derive the transactions from the DCM. The method is based on the use of the SLDNF proof procedure and can be implemented easily in Prolog.

We plan to extend our work in several directions. First, we would like to be able to simplify even more the transaction specifications. Second, we plan to consider the case of transactions consisting of multiple base facts, instead of just a single one. Finally, it might be convenient to consider the design and implementation of transactions in conventional architectures. In this sense, our current transaction must be seen as a transaction specification.

Finally, we would like to build a library of rule schemes, such that a particular DCM, and its corresponding IEM, could be developed by reusing the components of the library.

Acknowledgements

The authors wish to thank D. Costal, C. Martin, E. Mayol, J.A. Pastor, C. Quer, J. Sistac, E. Teniente and T. Urpí for their comments and suggestions on earlier drafts.
This work has been supported by the CICYT PRONTIC program, project TIC 680.

References

[Bub86] Bubenko,J.A. "Information system methodologies - A research view". In [OSV86], pp. 289-318.

[BuO86] Bubenko,J.A.;Olivé,A. "Dynamic or temporal modelling?. An illustrative comparison". SYSLAB Working Paper No. 117, University of Stockholm, 1986.

[CKM91] Chung,L.;Katalagarianos,P.;Marakakis,M.;Mertikas,M.;Mylopoulos,J.; Vassiliou,Y. "From information system requirements to design: A mapping framework", Information Systems, Vol.16, No.4, 1991, pp. 429-461.

[Fre85] Frenkel,K.A. "Toward automating the software development cycle". Comm. of the ACM, Vol.28,No.6,June 1985, pp. 578-589.

[MBJ90] Mylopoulos,J.;Borgida,A.;Jarke,M.;Koubarakis,M. "Telos: Representing knowledge about information systems", ACM Trans. on Information Systems, Vol.8,No.4, Oct.1990, pp 325-362.

[GMN84] Gallaire,H.;Minker,J.;Nicolas,J-M. "Logic and databases: A deductive approach". ACM Computing Surveys, Vol. 16, No. 2, June 1984, pp. 153-185.

[LIS91] Lloyd,J.;Shepherdson,J.C. "Partial evaluation in logic programming". J. Logic Programming, 1991, No.11, pp. 217-242.

[LIT84] Lloyd,J.W.;Topor,R.W. "Making Prolog more expressive". J. Logic Programming, 1984, No.3, pp. 225-240.

[GKB82] Gustafsson,M.;Karlsson,T.;Bubenko,J.A. "A declarative approach to conceptual information modelling". In [OSV82], pp. 93-142.

[MSS92] Mayol,E.; Sancho,M.R.; Sistac,J. "The ODISSEA project: An environment for the development of Information Systems from Deductive Conceptual Models". Proc. Intl. Workshop on the Deductive approach to IS and DB, Roses (Catalonia), 1992, pp. 17-47.

[Oli82] Olivé, A. "DADES: A methodology for specification and design of information systems". In [OSV82], pp. 285-334.

[Oli86] Olivé,A. "A comparison of the operational and deductive approaches to conceptual information systems modelling". Proc. IFIP 86, North-Holland, Dublin, 1986, pp. 91-96.

[Oli89] Olivé,A. "On the design and implementation of information systems from deductive conceptual models". Proc. of the 15th. VLDB, Amstedam, 1989, pp. 3-11.

[OSV82] Olle,T.W.;Sol,H.G.;Verrijn-Stuart,A.A. (Eds.) "Information systems design methodologies: A comparative review". North-Holland, 1982.

[OSV86] Olle,T.W.;Sol,H.G.;Verrijn-Stuart,A.A. (Eds.) "Information systems design methodologies: Improving the practice". North-Holland, 1986.

[San90] Sancho,M.R. "Deriving an internal events model from a deductive conceptual model". Proc. Intl. Workshop on the Deductive approach to IS and DB, S'Agaró (Catalonia), 1990, pp. 73-92.

[San92] Sancho,M.R. "The ODISSEA approach to the design of Information Systems from deductive conceptual models", Proc. of the IFIP World Congress, Madrid, September 1992, vol. 3, pp. 182-188.

[San93] Sancho,M.R. "Deriving transactions from deductive conceptual models", PhD. thesis. In preparation.

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Research Reports – 1994

LSI–94–1–R "Logspace and logtime leaf languages", Birgit Jenner, Pierre McKenzie, and Denis Thérien.

LSI–94–2–R "Degrees and reducibilities of easy tally sets", Montserrat Hermo.

LSI–94–3–R "Isothetic polyhedra and monotone boolean formulae", Robert Juan-Arinyo.

LSI–94–4–R "Una modelización de la incompletitud en los programas" (written in Spanish), Javier Pérez Campo.

LSI–94–5–R "A multiple shooting vectorial algorithm for progressive radiosity", Blanca Garcia and Xavier Pueyo.

LSI–94–6–R "Construction of the Face Octree model", Núria Pla-Garcia.

LSI–94–7–R "On the expected depth of boolean circuits", Josep Díaz, María J. Serna, Paul Spirakis, and Jacobo Torán.

LSI–94–8–R "A transformation scheme for double recursion", José L. Balcázar.

LSI–94–9–R "On architectures for federated DB systems", Fèlix Saltor, Benet Campderrich, and Manuel García-Solaco.

LSI–94–10–R "Relative knowledge and belief: SKL preferred model frames", Matías Alvarado.

LSI–94–11–R "A top-down design of a parallel dictionary using skip lists", Joaquim Gabarró, Conrado Martínez, and Xavier Messeguer.

LSI–94–12–R "Analysis of an optimized search algorithm for skip lists", Peter Kirschenhofer, Conrado Martínez, and Helmut Prodinger.

LSI–94–13–R "Bases de dades bitemporals" (written in Catalan), Carme Martín and Jaume Sistac.

LSI–94–14–R "A volume visualization algorithm using a coherent extended weight matrix", Daniela Tost, Anna Puig, and Isabel Navazo.

LSI–94–15–R "Deriving transaction specifications from deductive conceptual models of information systems", María Ribera Sancho and Antoni Olivé.

Copies of reports can be ordered from:

Nuria Sánchez
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Pau Gargallo, 5
08028 Barcelona, Spain
secrelsi@lsi.upc.es