Theoretical Computer Science 78 (1991) 113-136
North-Holland

The semantics of incomplete databases as an expression of preferences*

Véronique Royer

ONERA-CERT Departement d'Informatique, 2 avenue E. Belin B.P. 4025, 31055 Toulouse cedex, France

Abstract

Royer, V., The semantics of incomplete databases as an expression of preferences, Theoretical Computer Science 78 (1991) 113-136.

We address the problem of incomplete information in deductive databases from a semantic point of view. We want to treat the problem in a homogeneous way, using a formalism which can handle different types of incompleteness. We argue that it is convenient to define an incomplete database as a double entity: an underlying incomplete database together with selection criteria, formally some preference relation on the set of models of the underlying database, intended to reduce the incompleteness. Such an idea has been already exploited in the Perfect Model semantics of Negation as Failure in the stratified databases. We propose another notion of preference which induces a stronger selection of models and captures an intuitive process of "making choices" in disjunctive databases. We study both the declarative and the operational semantics for this notion of preference.

1. Introduction

1.1. Motivating the preference

The problem of incomplete information in knowledge representation systems is an important and challenging problem. The problem is particularly intricate because it often involves non-monotonic reasoning. We advocate in this section that theoretical issues concerning the study of non-monotonic logics can help to design a general framework for the treatment of incomplete information in knowledge or databases. The idea that one could formalize the treatment of incomplete information by means of preference relations stems from the work about the *preferential logics* pioneered in [29].

It is difficult to give a precise account of the notion of incompleteness without defining formally a knowledge or database. In this paper we take the viewpoint that a knowledge base is represented by a logical theory, in a very general sense, that

* This work was partially supported by the C.E.C. ESPRIT Project ESTEAM 316 and a project PRC BD3.

is by some triple $(L, T, \models)$ where L is a (first-order) logical language, T a set of formulas of L and $\models$ a particular entailment relation[1] defined on the formulas of L. In this setting, a knowledge base is incomplete if there is some formula ϕ of L such that $T \not\models \phi$ and $T \not\models \neg\phi$.

Non-Horn theories are simple examples of incomplete knowledge bases, due to the presence of disjunctive information. For example $\{A \vee B\}$ is incomplete with respect to the semantics of first-order logic: neither A nor $\neg A$ are logical consequences of $A \vee B$.

Incompleteness is a common feature in most formalisms because it is in general impossible to represent explicitly all the knowledge. Typically a knowledge representation formalism provides some conventions or rules to reduce, at least partly, the potential incompleteness situations. Most of the rules proposed in the literature share a common principle, which we like to call a "negation by default" principle: unless A can be derived, for some entailment or provability relation, $\neg A$ belongs to the knowledge base.

In logic programming the technique used to reduce incompleteness is the well-known Negation as Failure (NaF) rule, defined in [9] with respect to the procedural (SLD resolution) semantics of logic programs. It has a convenient implementation in the case of stratified programs with respect to the least fixed point semantics of [1].

In deductive databases or first-order theories, the negation is often treated through several types of "Closed World Assumption", that is completion of the initial theory under some "negation by default" rules: for example the GCWA rule [19], more recently the ECWA rule for circumscriptive theories and the ICWA rule for prioritized circumscription [11].

Concerning the logical foundations of the above rules, it is interesting to observe that the declarative (model-theoretic) semantics proposed in the literature also share a common principle, the *preferential logic* principle [5, 29]. The intended semantics can be expressed by preferring some models over others in the underlying theory. Formally, the general paradigm, formulated originally in [29] and generalized in [5], is that a non-monotonic logic should be expressed as a *preferential logic* $L_{\sqsubset} = (L, \sqsubset)$ composed of a monotonic logic L and a preference relation $\sqsubset$ (a partial order in [29], more generally any reflexive transitive relation in [5]) defined on the set of models of L. The models of $L_{\sqsubset}$ are the preferred models of L, that is the models of L which are maximal elements of $\sqsubset$. The different negation by default rules mentioned previously can be formalized as preferential logics. We briefly recall the essential ideas.

- The semantics of the GCWA rule is the minimal model semantics of first-order theories: the models which minimize the extensions of all the predicates are preferred to the standard Herbrand models [19]. Similarly, the semantics of circumscription (which gives a declarative semantics to the ECWA rule [11]) is

[1] Not necessarily the classical entailment relation of first-order logic.

obtained by selecting the models according to some preference criterion, or pre-emption ordering [3], which consists of minimizing the extensions of some predicates when some others may vary [15, 17].

- For the NaF rule, the formal preference relation describing the selection of models is the "preferability relation" defined in the Perfect Model semantics [22, 23]. It will be discussed in Section 3. The intuitive idea is to treat a rule of the form $\neg A \rightarrow B$ as the clause $A \vee B$ where B will be preferred to A. The incompleteness due to $A \vee B$ is reduced by selecting the model $\{B\}$ among the set of minimal models of $A \vee B$.

Not any non-monotonic logic can be defined as a preferential logic [18].[2] Nevertheless Shoham's paradigm is extremely interesting because it provides a unified and simple framework for the specification and comparison of non-monotonic logics. This is one motivation for trying to promote the approach in the particular case of incomplete knowledge or databases. The precedent examples of negation by default rules indicate that such an approach is a very natural one.

Consequently, a natural framework to deal with incomplete information in knowledge bases should be a formalism addressing both problems:

- the definition of the *underlying incomplete data or knowledge base*,
- the specification of the *control of incomplete information*, that is the particular techniques used to remove or reduce incompleteness.

The adaptation of Shoham's ideas to the treatment of incomplete information results in the following paradigm: *the control of incomplete information should be described by a preference relation on the models of the underlying incomplete data or knowledge base.*

1.2. Directions of work

We want to develop and to validate these ideas for *incomplete deductive databases.* Two points must be examined to show that the above paradigm is suitable for a general treatment of incomplete databases.

- Whether the paradigm is general enough to formalize other types of control than those based on negation by default rules.
- Whether the paradigm can support a computational treatment of incomplete information. It is not sufficient to design the semantics of incomplete databases in terms of a preferential semantics (that is as a set of models maximizing some preference relation). One has furthermore to develop algorithms calculating the preferred models. This is the problem of finding an equivalent "operational semantics", in the standard software engineering terminology [31].

In this paper we will define another type of control for incomplete information which is not a negation by default principle. It corresponds to an intuitive process of *making choices in disjunctive databases* according to some preference criteria expressed on the atoms of the database: "if there is a possible choice between A

[2] Actually Default Logic [25] cannot be expressed as a preferential logic.

and B and if B is preferred to A, then A will be rejected". We will define a preferential semantics for "Making Choices". Though the Making Choices control has been introduced essentially for theoretical purposes, we think that the concept is of practical interest and discuss possible applications in the conclusion.

For the second point, very little work has been done to study the operational properties of the preferential logics proposed in the literature on non-monotonic reasoning. Shoham does not address this point at all in his work on the logic of Chronological Ignorance [30]. The computation of Circumscription by MILO Resolution appears rather unsatisfactory and inefficient [24]. Even for the Perfect Model semantics of Negation as Failure in stratified databases the computational aspects have received little attention. The only work addressing the case of stratified programs (that is rules allowing no disjunction in the consequences) appears in [12].

We define here an operational semantics for stratified databases which extends naturally the classical least fixed point semantics proposed in the literature for the stratified programs [1] and for the disjunctive databases [20]. We also study the operational properties of the "Making Choices" preferential logic. We show that the Making Choices semantics is calculable when the preference criteria for making choices are not contradictory (acyclic preference relations) and define a least fixed point semantics. The more general problem of query evaluation is not treated here, though one can expect that the least fixed point semantics proposed in this paper can support efficient query evaluation strategies.

The plan of the paper is as follows. Section 2 gives some preliminaries about the declarative and least fixed point semantics of disjunctive databases. Section 3 presents the Perfect Model semantics as a preferential logic for disjunctive databases. We give a least fixed point semantics for the stratified databases, which is to our knowledge a new result in the literature. Section 4 is concerned with the definition of Making Choices semantics for a class of incomplete databases called "ordered databases". Finally the Perfect Model and the Making Choices semantics are compared in the last section. We conclude by emphasizing some advantages of the preferential formalism in general and of the Making Choices semantics in particular.

We assume the reader is familiar with the notions of Herbrand universe, Herbrand base and Herbrand model of a first-order theory. In the whole paper, "model" means "Herbrand model".

2. Preliminaries about disjunctive databases

Definition 2.1. A *disjunctive database* is a finite set of *disjunctive rules* that is (implicitly closed) formulae of the form $A_1 \wedge \cdots \wedge A_n \rightarrow B_1 \vee \cdots \vee B_m$, where the A_i and B_j are all atomic formulae, $n \geqslant 0$, $m > 0$.

For the sake of brevity we write $A_1 \ldots A_n \rightarrow B_1 \ldots B_m$ for the rule $A_1 \wedge \cdots \wedge A_n \rightarrow B_1 \vee \cdots \vee B_m$. We call a *disjunctive fact* any ground (i.e. variable free) instance of a database rule having no premise ($n = 0$).

We consider that the natural declarative semantics of a disjunctive database is defined by its set of minimal Herbrand models, that is the Herbrand models of the database (considered as a first-order theory) which are minimal with respect to set inclusion. Minimality means "no junk": the informative content of a database is just the minimal information deducible from the database facts and rules. Herbrand model means "no confusion": two distinct symbols denote two distinct entities.

An operational semantics has to give an alternative and constructive characterization of the declarative semantics. For definite Horn databases the fixed point semantics of [13] fits well: the fixpoint operator, called "immediate consequence" operator, simulates the forward chaining execution of the database rules and its least fixpoint represents the content of the database, that is the ground atomic formulae deducible from the database.

For disjunctive databases, we have to switch from the generation of atomic formulae to the generation of disjunctive positive formulae. The notion of disjunctive consequence deducible from the database is captured by the notion of *characteristic clause.*[3]

Definition 2.2. A *characteristic clause* of a disjunctive database DDB is a positive ground clause which is a logical consequence of DDB and is minimal for subsumption, that is: $\mathrm{DDB} \models \alpha$ and there is no positive clause α' such that $\alpha' \subset \alpha$ and $\mathrm{DDB} \models \alpha'$.[4,5] The *characteristic set* of a disjunctive database DDB is the set of characteristic clauses of DDB and will be denoted by Char(DDB).

The characteristic set of a disjunctive database DDB provides a representation of the set of minimal models of DDB as follows.

Proposition 2.3. *A Herbrand structure is a minimal model of a disjunctive database DDB iff it is a minimal Herbrand model of its characteristic set* Char(DDB).

A further interest of the characteristic set Char(DDB) is that it can be calculated as the least fixed point of a monotonic operator generalizing the immediate consequence operator of definite databases [13]. The forward chaining execution of the disjunctive database rules generates sets of (ground positive) clauses instead of sets of atoms. This changes the notion of evaluation of the rule premises with respect to the current computation state, actually a set of clauses representing the current approximation of the set of models to be constructed. Intuitively a rule $A \rightarrow \beta$ will be fired when there is some clause $A \vee \alpha$ in the current computation state and will produce the new clause $B \vee \beta$. The underlying process is the hyperresolution rule of [8] (positive hyperresolution with atomic electrons). The above remarks result

[3] The notion was introduced in a more general setting in [6].

[4] $\models$ denotes here the entailment relation of first-order logic.

[5] $\subset$ denotes here the usual subsumption relation between clauses (considered as sets of literals with no repetition) [8].

in the following least fixed point semantics for disjunctive databases. The result has already appeared in [27] and in a slightly different form in [10, 20].

Proposition 2.4 (Least fixed point semantics for disjunctive databases). *Let DDB be some disjunctive database. Let* T_{DDB} *be the following operator, called the* immediate consequence operator *for DDB, defined on the power set of the set of ground positive clauses formed by Herbrand atoms of DDB*:

$$T_{\mathrm{DDB}}(I) = I \cup \{B_1 \vee \cdots \vee B_p \vee \alpha_1 \cdots \vee \alpha_n, \text{ such that there exists a ground instance of a rule in DDB: } A_1 \ldots A_n \rightarrow B_1 \ldots B_p, \text{ and } \forall i \in [1, n], A_i \vee \alpha_i \in I\}.$$

Let μT_{DDB} *be defined as the following union:* $\bigcup_{i=1}^{i=\infty} T^i_{\mathrm{DDB}}(\emptyset)$. *Then the next three points hold*:

- T_{DDB} *is a monotonic operator* (*for set inclusion*) *and* μT_{DDB} *is its least fixed point* (*Knaster-Tarsky theorem*).
- *A model is a minimal model of DDB iff it is a minimal model of* μT_{DDB}.
- Char(DDB) = subsume(μT_{DDB}), *where "subsume" is the elimination operation of subsumed clauses.*

Example 2.5. DDB = $\{A \vee B, A \rightarrow C.D, B \rightarrow E.F, E \rightarrow G\}$. The characteristic set of DDB is $\{A \vee B, B \vee C \vee D, A \vee E \vee F, C \vee D \vee E \vee F, A \vee G \vee F, C \vee D \vee G \vee F\}$. Adding the new fact $D \vee E$ gives the clause $D \vee G$ subsuming the clause $C \vee D \vee G \vee F$. Hence the characteristic set becomes

$$\{A \vee B, B \vee C \vee D, A \vee E \vee F, C \vee D \vee E \vee F, A \vee G \vee F, D \vee G\}.$$

In the case the Herbrand universe is finite (no function symbol in the database), a more efficient computation of Char(DDB) is possible when the subsumed clauses are eliminated immediately after each iteration of the operator T_{DDB}. This gives rise to a new operator, we call the *characteristic operator* for DDB: $\mathcal{T}_{\mathrm{DDB}} = \text{subsume} \circ T_{\mathrm{DDB}}$. $\mathcal{T}_{\mathrm{DDB}}$ is no longer monotonic (the subsume operation is non-monotonic), but it still has a least fixed point which represents Char(DDB). One easily shows that there exists some integer N such that $\mathcal{T}^N_{\mathrm{DDB}}(\emptyset)$ is a least fixed point of $\mathcal{T}_{\mathrm{DDB}}$ and Char(DDB) = $\mathcal{T}^N_{\mathrm{DDB}}(\emptyset)$.

3. Perfect Model semantics for incomplete databases

Perfect Model semantics was initially presented to give a declarative semantics for the class of stratified logic programs [22]. It was extended to the larger class of stratified databases, that is sets of rules with positive or negated premises and disjunctive consequences [23] such that there is no cyclic predicate definition involving negations. For this class the Perfect Model semantics captures the natural semantics of Negation as Failure. Perfect Model semantics goes into the general

framework presented in the introduction. Indeed a stratified database can be represented by:

- on the one hand, a disjunctive database: the underlying incomplete database, obtained by rewriting the original rules by moving the negative premises from the left to the right of the implication symbol;
- on the other hand, a "priority" relation defined on the Herbrand base expressing the particular negation as failure status of the negated premises.

The preferential logic corresponding to Perfect Model semantics is given by the particular preference relation on the set of minimal models of the underlying disjunctive database which is associated with the priority relation.

The notion of priority allows us to capture some "procedural" interpretation of the database rules. The symbol "$\rightarrow$" in a rule is not the pure logical implication, but should rather be interpreted as a procedure declaration. For example the procedural interpretation of the rule $A.\neg B \rightarrow C.D$ is that C or D will be produced only after A and $\neg B$ have been derived. When the rules have only positive premises, the semantics of minimal models is sufficient to capture this interpretation. It is no longer sufficient when we have negative premises evaluated by failure. Indeed $\{A, B\}$ is a minimal model of the set of first-order formulas $\{A, A \wedge \neg B \rightarrow C\}$ but does not reflect the intended semantics of the *database* $\{A, A.\neg B \rightarrow C\}$. The intended model is $\{A, C\}$ which is *preferable* to $\{A, B\}$ due to the negation by failure status of B. It is as if we have assigned to C a greater preference than B $(B < C)$ which allows us to select the model $\{A, C\}$ among the set of minimal models of the disjunctive database $\{A, A \rightarrow B.C\}$. The difficulty is to formalize these intuitive ideas by means of a formal model preference relation.

3.1. Perfect Model semantics as a preferential logic

We recall the essential results concerning Perfect Model semantics. Our presentation differs slightly from the original formulation of [22, 23] just because we want to emphasize the "preferential" nature of Perfect Model semantics. The terminology "non-definite" is chosen to contrast with the term "definite" used for "definite" databases, that is sets of definite Horn rules [13].

Definition 3.1. (1) A *non-definite database* is a finite set of rules written $A_1 \ldots A_n \rightarrow B_1 \ldots B_m$ where the A_i are *literals* (atomic or negations of atomic formulae) and the B_j are *atomic* formulae.

(2) The *disjunctive database associated to a non-definite database* NDB is defined as the set of all the rules $A1 \ldots A_n \rightarrow B1 \ldots Bm.C1 \ldots Cp$ such that there is a rule of NDB of the form $A1 \ldots An.\neg B1 \ldots \neg Bm \rightarrow C1 \ldots Cm$ (where the A_i, B_j and C_k are all atomic formulae).

(3) The *priority relation* of a non-definite database NDB, denoted $<$,[6] is the least binary relation on the Herbrand base of NDB verifying:

[6] Denoted $>$ in [22, 23].

- for any ground instance of a rule in NDB $A1 \ldots An.\neg B1 \ldots \neg Bm \rightarrow C1 \ldots Cp$: $Bi < Cj$ for any $i = 1 \ldots m$ and $j = 1 \ldots p$;
- $<$ is transitive,
- $<$ is closed by "forward chaining": if $A < B$ and there is a ground instance of a rule of the form $\ldots (\neg) B \ldots \rightarrow \ldots C \ldots$, then $A < C$,
- $<$ is closed by "backward chaining": if $B < C$ and there is a ground instance of a rule of the form $\ldots (\neg) A \ldots \rightarrow \ldots B \ldots$ then $A < C$.

(4) The priority relation induces a *preferability relation*, denoted $\ll$, defined on the minimal models of the underlying disjunctive database as follows. $M \ll N$[7] iff for every atom n of $N \setminus M$ there is an atom m of $M \setminus N$ such that $m < n$.

(5) A *perfect model* of a non-definite database is a minimal model of its disjunctive database which is maximal for the preferability relation $\ll$.

In the presentation of [22, 23], the preferability relation $\ll$ is defined more generally on the set of Herbrand models (not necessarily minimal) of the underlying disjunctive database. However one easily shows that the perfect models are minimal models. Our presentation stresses the fact that Perfect Model semantics is a preferential semantics based on the minimal model semantics of disjunctive databases.

There are sufficient conditions for the existence of perfect models, the so-called *stratifiability conditions*. Intuitively they state that a predicate cannot depend recursively from its own negation.

By definition a non-definite database is *stratified* [1, 23] iff there is a partition $\bigcup_{i=1}^{i=N} P_i$ of the set of predicates symbols of the database such that for every rule, whenever P is the predicate symbol of some negated premise and Q is the predicate symbol of some consequent of the rule, if P belongs to P_i then Q belongs to some P_j with $i < j$. A more general condition is the *local stratification* condition [23]. By definition, a non-definite database is said to be *locally stratified* iff there is an integer function f defined on the Herbrand base of the database, such that for every ground instance of a rule of the form $A_1 \ldots A_n.\neg B_1 \ldots \neg B_m \rightarrow C_1 \ldots C_p$ we have:

- $\forall k \in [1, p]\ f(C_1) = f(C_k)$,
- $\forall i \in [1, n]\ f(A_i) \leqslant f(C_1)$,
- $\forall j \in [1, m]\ f(B_j) < f(C_1)$.

It is easy to verify that the local stratification condition is equivalent to the non-cyclicity of the priority relation [23]. This guarantees the existence of a perfect model.

Proposition 3.2 (Przymusinski [22], existence of perfect models). *Any locally stratified database (thus a fortiori any stratified database) has a (at least one) perfect model.*

The above property is lost when the stratification condition fails. For example the database $\{\neg A \rightarrow B, \neg B \rightarrow A\}$ has no perfect model (the cycle $A < B < A$ makes the preferability relation cyclic).

[7] $N < M$ in [22, 23].

3.2. Least fixed point semantics for the stratified databases

NB: From now on, without more precision, "stratification" means "local stratification".

We address now the problem of the computation of Perfect Model semantics. More precisely we try to extend the least fixed point semantics approach, which works well for the disjunctive databases (cf. Section 2) and the stratified programs [1], to the class of stratified databases. Such an extension is possible. The price to be paid is the introduction of the class of "clause rules" which is more complex than the class of disjunctive databases. The resulting computation process for the perfect models becomes also more complicated.

We first recall some well-known results. For the stratified *logic programs*, that is sets of rules having unit consequents, there is a *unique perfect model* which coincides with the "iterated least fixpoint model" defined in [1]. The resulting operational semantics is traditionally called a *least fixed point semantics.* Precisely, the perfect model M of a given locally stratified logic program SP is calculated as the union (limit of the sequence) of the least fixed point models M_i obtained for each stratum S_i of the program.

Proposition 3.3 (Apt et al. [1], least fixed semantics of stratified programs). *Let* SP *be some stratified logic program, and f a stratification function for* SP. *Let* $\mathrm{SP} = \bigcup_{i=1}^{i=N} S_i$ *be the partition of* SP *into strata, where the stratum* S_i *is the set of the* (*ground instances of*) *rules of* SP *whose consequents have the common level i* (*with respect to f*). *Then the perfect model M of* SP *is equal to* M_N,

- *where for every* $i \in [1, N]$, $M_i = \bigcup_{j=1}^{j=\infty} T_i^j(M_{i-1})$,
- *where* T_i *is the immediate consequence operator associated with the stratum* S_i, *i.e.*

$$T_i(E) = \{C, \exists A_1 \ldots A_n . \neg B_1 \ldots \neg B_m \rightarrow C \in S_i \text{ such that } \forall j \in [1, n], A_j \in E \text{ and } \forall k \in [1, m], B_k \notin M_{i-1}\}.$$

For stratified databases, the problem is to accommodate the disjunctive and negative information together. For disjunctive databases the least fixed point semantics consists of generating a set of "characteristic clauses" representing the set of minimal models of the database. We are looking for a similar representation of the set of perfect models. For stratified databases a set of models will be constructed at each stratum S_i. They will be represented by a set of clauses, say C_i. The evaluation of negated atoms in a level $i+1$ rule will be done with respect to C_i.

The key idea is that any ground instance of a level $i+1$ rule of the form $A_1 \ldots A_n . \neg B_1 \ldots \neg B_m \rightarrow \gamma$ (where γ is a positive clause) can be rewritten into:

- the rule $A_1 \ldots A_n . \neg B_2 \ldots \neg B_m \rightarrow \gamma$, if the atom B_1 does not appear in any clause of C_i (B_1 is an ECWA atom in the sense of [11]),
- or the rule $A_1 \ldots A_n . \beta_1 . \neg B_2 \ldots \neg B_m \rightarrow \gamma$, if the atom B_1 appears in some *non-unit* clause of C_i, and β_1 is the *residue* of B_1 in C_{i-1} *that is the conjunction of all* (*non-empty*) *clauses* β' such that $B_1 \vee \beta'$ is a clause of C_{i-1}.

The previous idea suggests some rewriting of the ith stratum to replace the negated atoms by their residues according to the lower level strata. In this process the negative information is replaced by a positive one. The resulting database we will call *a clause base*, that is a finite set of rules of the form: $\alpha_1 \ldots \alpha_n \rightarrow \gamma$, where the α_i and γ are positive clauses (disjunctions of atomic formulae). The final remark is that the above rewriting works only if the residues are finite conjunctions. This means that the sets C_i must be finite. This will be the case if the given database has a finite Herbrand universe, which is the case if there is no function symbol in the rules.

3.2.1. Clause bases

We do not develop the discussion about clause bases here (we intend to do this elsewhere). We just indicate that the clause bases represent the obvious extension of disjunctive databases when the premises of the rules are generalized to positive clauses. The declarative semantics of clause bases is again the minimal model semantics (that is the set of minimal models of the underlying first-order theory).

The least fixed point semantics of disjunctive databases extends very naturally to the clause bases. The only significant change concerns the evaluation of the rule premises with respect to the current set of clauses. One must look for clauses having a non-empty intersection with the premise clauses in order to fire a given rule. The computation process is positive hyperresolution [8], this time in its full generality.

Definition 3.4 (*Immediate consequence operator for clause bases*). Let CDB be some clause base. The immediate consequence operator for CDB, T_{CDB}, is defined on the power set of the set of ground positive clauses formed by Herbrand atoms of DDB, as follows:

$$T_{\mathrm{CDB}}(I) = I \cup \{\gamma \vee \gamma_1 \vee \cdots \vee \gamma_n \text{ such that there is a ground instance of a rule in CDB: } \alpha_1 \ldots \alpha_n \rightarrow \gamma, \text{ for every } i \in [1, n] \text{ there is a clause } \beta_i \text{ in } I \text{ such that } \alpha_i \cap \beta_i \neq \emptyset \text{ and } \gamma_i = \beta_i \backslash \alpha_i\}.^{8}$$

Proposition 3.5 (Least fixpoint semantics for clause bases). *Let* Char(CDB), *called the characteristic set of the clause base CDB, be defined as* subsume(μT_{CDB}) *where* μT_{CDB} *is the least fixed point of* T_{CDB}. *A Herbrand structure of CDB is a minimal model of CDB iff it is a minimal model of its characteristic set.*

3.2.2. Operational semantics for function free stratified databases

Definition 3.6 (*The clause base associated with a stratified database*). Let SDB be some *function free* stratified database and $\mathrm{SDB} = \bigcup_{i=1}^{i=N} (S_i)$ its decomposition into

[8] Denotes the set difference operation, that is γ_i is the clause obtained by eliminating from β_i the atoms common with α_i.

strata (every fact of SDB belongs to the first stratum). The clause base CDB associated with SDB is defined as follows:

- $\mathrm{CDB} = \bigcup_{i=1}^{i=N} \mathrm{CDB}_i$.
- $\mathrm{CDB}_0 = S_0$.
- $\mathrm{CDB}_i = \mathrm{CDB}_{i-1} \cup \mathrm{rewrite}(S_i, C_{i-1})$, for every $i \in [1, N]$.
- $C_i = \mathrm{Char}(\mathrm{CDB}_i)$, for every $i \in [1, N]$.
- If S is a set of non-definite rules and C a set of positive ground clauses, then rewrite(S, C) is the set of clause rules obtained as follows: for any ground instance of a rule in S of the form $A_1 \ldots A_n.\neg B_1 \ldots \neg B_m \rightarrow \gamma$, if no B_i is a unit clause in C then rewrite(S, C) contains the clause rule

$$A_1 \ldots A_n.\mathrm{res}(B_1, C) \ldots \mathrm{res}(B_m, C) \rightarrow \gamma.$$

- For any atom B and any finite set of positive ground clauses C, the residue of B in C, denoted by res(B, C), is the conjunction of every clause $\beta \backslash B$ such that $B \in \beta$ and $\beta \in C$. The residue is the distinguished atom "true" if C contains no clause with B.

Proposition 3.7 (Least fixed point semantics of stratified databases). *Let SDB be some function free stratified database, and CDB its associated clause base. Then the perfect models of SDB are exactly the minimal models of CDB. Moreover the perfect models of SDB are the minimal models of the set* Char(SDB), *called the* characteristic set of *SDB, defined by* $\mathrm{Char}(\mathrm{SDB}) = D_N$ *where*

- $D_0 = \emptyset$,
- $\forall i \in [1, N]$, $D_i = \mathrm{subsume}(T^{\omega}_{\mathrm{CDB}_i}(C_{i-1}))$.[9]

Example 3.8. $\mathrm{SDB} = \{A \vee B, B \vee D, \neg A \rightarrow C, \neg B \rightarrow C\}$. The perfect models of SDB are $\{A, D, C\}$ and $\{B, C\}$. The model $\{A, B\}$ of the underlying disjunctive database is not perfect (less preferable than $\{B, C\}$). The clause base associated with SDB is $\mathrm{CDB} = \{A \vee B, B \vee D, B \rightarrow C, A.D \rightarrow C\}$. In this case we get a simple disjunctive database. The characteristic set of CDB is subsume$(\{A \vee B, B \vee D, A \vee C, D \vee C, B \vee C, C\})$ that is $\{A \vee B, B \vee D, C\}$. This gives effectively a representation of the perfect models of SDB.

4. Making Choices in ordered databases

We now present another way of controlling incomplete information which corresponds to some process of *making choices* in disjunctive databases. We show how to formalise this new type of treatment of incomplete information with the paradigm presented in the introduction.

[9] With the standard definition of $T^{\omega}(I) = \bigcup_{i=1}^{i=\infty} T^i(I)$.

We introduce the concept of "ordered database" as a disjunctive database together with some binary relation defined on the Herbrand base and called a preference relation. The formal preferential semantics of "Making Choices" will be fixed by defining what kind of selection the atomic preference induces into the set of models of the underlying disjunctive database. Hence a first problem is to extend the given atomic preference into a model preference relation capturing the intuition of Making Choices. The second problem is to keep the concept of Making Choices tractable. The solution we propose here has been defined in order to satisfy this condition. The declarative semantics of Making Choices supports an effective operational semantics, less complex than the operational semantics of Negation as Failure presented in Section 3.

4.1. The preferred models of ordered databases

We present the declarative semantics of "Making Choices" for ordered databases. The intended models of an ordered database will be defined as the models of the underlying disjunctive database which are the maximal elements of the model preference relation. They will be called "preferred".

Definition 4.1. An *ordered database* (ODB in short) is defined by a *disjunctive database and a strict partial order*, called the *atomic preference*, noted $\prec$, defined on the Herbrand base of the database.

There are several ways to induce a relation among sets from a relation among atoms. Here we cannot simply consider a "quantitative" preference criterion, for example by defining the higher preference models to be those containing the maximum number of atoms of higher preference. The difficulty is to take into account the qualitative features of the problem, namely that the considered sets are actually models of a deductive database. Thus the model preference relation will be based on the following principles:

- If $A \vee B$ and $C \vee D$ are both minimal consequences of the underlying database, if C is preferred to A, then no model with A cannot be preferred. It is as if A has been eliminated completely from the database.
- The derivation of atoms of higher preference is forced, whether or not they do effectively participate in some choice. This means to relax the constraint in the definition of perfect models: "$\forall n \in N \setminus M$, $\exists m \in M \setminus N$ such that $m \prec n$", to a weaker condition stating that an atom n of $N \setminus M$ needs to be justified by a less preferred atom in $M \setminus N$ only if there are atoms preferred to n in M.
- The preference between models is compatible with the natural forward chaining execution of the database rules. Whenever A is preferred to B, any possible atom derivable from A and the database rules must be preferred to B. Hence the model preference is defined, not from the original atomic preference, but from its "*forward chaining*" *closure.*

Definition 4.2 (*The forward chaining closure of the atomic preference*). Let ODB = (ODB, $\prec$) be some ordered database. The forward chaining closure of the relation $\prec$ is the least relation $\prec_*$ containing $\prec$ and closed by the law: if $A \prec_* B$ and there is a rule instance in DDB of the form $\ldots B \ldots \rightarrow \ldots C \ldots$, then $A \prec_* C$.

Definition 4.3 (*Model preference relation for Making Choices*). Let ODB = (DDB, $\prec$) be an ordered database. The model preference relation for ODB, noted $\ll$ is defined as follows. For every model M and N of DDB, $M \ll N$ iff:

- *Justification condition*: $\forall n \in N \backslash M$, either $\exists m \in M \backslash N$ such that $n \prec_* m$, or $\exists m \in M \backslash N$ such that $m \prec_* n$.
- *Strictness condition*: there is at least $m \in M \backslash N$ and $n \in N$ such that $m \prec_* n$.

Definition 4.4 (*Preferred models*). The *preferred models* of an ordered database (DDB, $\prec$) are the minimal models of DBB which are maximal elements of the model preference relation $\ll$.

We now illustrate the definitions by some examples.

Example 4.5 (*Illustration of the strictness condition*). DDB = $\{A1 \vee B, A2 \vee C, A1 \rightarrow G, A2 \rightarrow F\}$ with $A1 \prec F$ and $A2 \prec G$. The minimal models are $M1 = \{B, C\}$, $M2 = \{A1, G, C\}$, $M3 = \{A2, F, C\}$ and $M4 = \{A1, A2, G, F\}$. $\prec_*$ is equal to $\prec$. The preference relation is: $M4 \ll M2$ and $M4 \ll M3$. Hence the preferred models are $M1$, $M2$, $M3$. In $M4 \ll M2$, we accept that G is justified by $A2$ though G is common to both models. This means to prefer a model with as few as possible lower preference atoms. Here the strictness condition is absolutely necessary because the justification condition is satisfied trivially for every pair of models. In the absence of the strictness condition all models become comparable, hence there would not be preferred models.

Example 4.6 (*Illustration of the closure condition for* $\prec_*$). BDD = $\{A \vee B, A \rightarrow C.D, B \rightarrow E.F, E \rightarrow G\}$ with $A \prec E$ and $G \prec C$. By forward chaining closure we get $A \prec_* G$. The models are $\{A, C\}$, $\{A, D\}$, $\{B, F\}$ and $\{B, E, G\}$. If the model preference was based only onto $\prec$, $\{A, C\}$ and $\{B, E, G\}$ become uncomparable. This seems to us not compatible with the natural behavior of the database. Indeed, as B allows the derivation of an atom preferred to A (E), B is chosen rather than A in $A \vee B$. But G comes with E and the preference $G \prec C$, where C depends on A, competes with the precedent choice $A \prec E$. Adding $A \prec_* G$ makes $G \prec C$ ineffective. Finally we get $\{A, C\} \ll \{B, E, G\}$ and $\{A, D\} \ll \{B, E, G\}$. Thus the preferred models are $\{B, F\}$ and $\{B, E, G\}$.

Example 4.7 (*No preferred model*). BDD = $\{A; A \rightarrow B.C, C \rightarrow D.E, E \rightarrow F\}$ with $B \prec C$, $D \prec E$ and $F \prec B$. The minimal models are $M1 = \{A, B\}$, $M2 = \{A, C, D\}$ and $M3 = \{A, C, E, F\}$. We have the cycle $M1 \ll M2 \ll M3 \ll M1$, hence there is no preferred model. The cycle situation is due to cycles in $\prec_*$ (not necessarily in $\prec$).

In the following we will show that the non-existence of cycle in $<_*$ is a sufficient condition to the existence of preferred models, when the Herbrand base is finite.

4.2. Making Choices as a preferential logic

The definition of ordered database presented here is not exactly the definition of a preferential logic in the sense of [29], where the preference relation between the models of the underlying logic must be a partial order, nor in the sense of [7] where the preference relation can be an irreflexive transitive relation. As a matter of fact the model preference relation $\ll$ is an irreflexive, not necessarily transitive relation.

Let us consider for example $\mathrm{DDB} = \{A \vee A' \vee A'', B \vee B' \vee B'', C \vee C' \vee C''\}$ with $B < B' < B''$ and $C < C' < C''$ and $A'' < A$. The models $M1 = \{A, B, C\}$, $M2 = \{A', B', C'\}$ and $M3 = \{A'', B'', C''\}$ verify $M1 \ll M2 \ll M3$ but we do not have $M1 \ll M3$.

However this is not a crucial problem. To respect the formulation of Shoham's paradigm simply means to consider that the preferential logic of Making Choices is the transitive closure of $\ll$. Indeed taking the transitive closure does not change the preferred models (the maximal elements of the relation $\ll$ and of its transitive closure are the same).

4.3. Operational semantics of strict ordered databases

In this section we define a calculable subclass of ordered databases. This is the class of the ordered databases with no function symbol and acyclic relation $<_*$, called strict ordered databases. The non-cyclity of $<_*$ allows an effective construction of preferred models in the case where the Herbrand base is finite. Precisely we will show that the preferred models of strict ordered databases can be represented by sets of clauses calculated:

- from the set of clauses representing the models of the underlying disjunctive database,
- by some elimination process of the less preferred atoms, called reduction.

Let us notice that the reduction process may be defined more generally for an ordered database allowing function symbols (but with non-cyclic $<_*$). However the reduction will be an infinistic and non-terminating process, if the graph of the preference relation is infinite.

4.3.1. Definition of characteristic clauses for strict ordered databases

Definition 4.8. A *strict ordered database* is an ordered database without function symbol such that the forward chaining closure $\prec_*$ of its preference relation is acyclic.

The set of *characteristic clauses of a strict ordered database* SODB, denoted also Char(SODB), will be defined from the characteristic set of its underlying disjunctive database DDB by the following *reduction process.*

Definition 4.9. (1) Let Δ be a set of ground positive clauses and $\prec$ a strict partial order on the atoms of Δ.

(2) An atom A of Δ is said reducible in Δ with respect to $\prec$ if it does not appear in Δ as unit clause and there is an atom B in a clause of Δ such that $A \prec B$.

(3) The *A-reduct* of Δ with respect to $\prec$ is:

- Δ, if A is not reducible in Δ with respect to $\prec$;
- otherwise, the set obtained by first eliminating A from the clauses of Δ, then eliminating the eventually subsumed clauses.

(4) Let $\alpha = (A1, A2, \ldots, An)$ be a list of atoms of Δ. The *α-reduct* of Δ is defined as the α_1-reduct of the $A1$-reduct of Δ, where $\alpha_1 = (A2, \ldots, An)$.

(5) A set of clauses is called *reduced* with respect to $\prec$ iff it contains no atom reducible with respect to $\prec$. A *complete reduction order* for Δ with respect to $\prec$ is a list $\alpha = (A1, \ldots, An)$ such that for no $i < j$ we have $Aj \prec Ai$, and the α-reduct of Δ is reduced with respect to $\prec$.

Now let us consider a strict ordered database SODB and its disjunctive database DDB. Let A be some minimal element of $\prec$ such that there exists B in Char(DDB) preferred to A. The A-reduct of Char(DDB) is a set of clauses representing models without A. In other terms, the A-reduction simulates the selection of models with atoms preferred to A. If we repeat the reduction process, in a way compatible with the preference hierarchy, one can expect to obtain clauses representing the preferred models.

However the reduction order is important. Indeed an A-reduction can forbid a B-reduction if in the first one the clauses containing all possible justifications for B have been eliminated. Similarly the (A, B)-reduction may be not equivalent to the (B, A)-reduction. In fact, the reduction orders must be *compatible with the closure* $\prec_*$[10] and all need to be considered.

Another point is that some preference couples $A \prec B$ may have no influence on the selection of models. This is exactly the case for the couples (A, B) where B is *strongly dependent* of A in SODB, which means that any minimal model of DDB containing A also contains B. Indeed such atoms A and B cannot satisfy the justification or strictness conditions in the definition of $\ll$. The "meaningless" couples $A \prec B$, where B is strongly dependent on A, must not participate in the reduction process otherwise we lose the completeness property stated below. Thus they must not be considered in the definition of the reduction process. Furthermore, as they do not contribute to the definition of the model preference relation, they can be simply eliminated from the atomic preference relation (or eventually from its forward closure). *In the following we implicitly assume that it is the case.* We conclude by briefly indicating how to test the strong dependency property.

The strong dependency property is equivalent to a simple syntactic property of the database. Intuitively an atom B is strongly dependent of A in a disjunctive

[10] Because the selection of models is based on $\prec_*$.

database DDB iff A belongs to every possible "defining set" for B, that is set of atoms involved in some derivation of B from the database rules. More formally, a "defining set for an atom B in a disjunctive database DDB" is recursively defined as any set S of atoms such that there is a ground instance of a rule of DDB $A1 \ldots An \rightarrow A \ldots$ and $S = \bigcup_{i=1}^{i=n} S_i$ where S_i is a defining set for A_i (if the rule has no premise $S = \{A\}$).

We can now state the formal results. All the proofs can be found in the appendix.

Proposition 4.10 (Validity of the reduction process). *Let SODB* = (DDB, $<$) *be a strict ordered database. Let* α *be a complete reduction order for* Char(DDB) *with respect to* $<_*$. *Then any minimal model of the* α*-reduct of* Char(DDB) *is a preferred model of SODB.*

Proposition 4.11 (Completeness of the reduction process). *For any preferred model M of SODB there is a complete reduction order* α *for* $<_*$ *such that M is a minimal model of the* α*-reduct of* Char(DDB).

4.3.2. Examples

The following examples illustrate how the reduction process works.

Example 4.5 (*Continued*). The characteristic set of the underlying disjunctive database is $\{A1 \vee B, A2 \vee C, G \vee B, F \vee C\}$. There are two possible reduction orders: $(A1, A2)$ or $(A2, A1)$. The first one gives the set of reduced clauses $\{B, A2 \vee C, F \vee C\}$ which represents the models $M1$ and $M2$. The second one gives the clauses $\{C, A1 \vee B, B \vee G\}$ representing the models $M1$ and $M3$.

Example 4.6 (*Continued*). The characteristic set of DDB is $\{A \vee B,\ B \vee C \vee D,\ A \vee E \vee F,\ C \vee D \vee E \vee F,\ A \vee G \vee F,\ C \vee D \vee G \vee F\}$. There is only one reduction order compatible with $<_*$: (A, G). We get the reduced clauses $\{B, E \vee F, G \vee F\}$. Doing the reduction in the reverse order results in the clauses $\{A \vee B, A \vee F, B \vee C \vee D, C \vee D \vee F\}$. One minimal model of these clauses is $\{A, B\}$ which is not a preferred model.

Example 4.12 (*Elimination of meaningless preferences*). DDB = $\{A \vee B, A \rightarrow C.D\}$ with $A < C$ which is meaningless because C strongly depends on A. The models are $\{B\}$, $\{A, C\}$ and $\{A, D\}$ and are uncomparable, hence both perfect. The characteristic set is $\{A \vee B, B \vee C \vee D\}$. However the A-reduction gives only the reduced clause $\{B\}$. Thus we lose the two other models.

4.3.3. Operational semantics for the strict ordered databases

We have defined the reduction process on the characteristic set of the underlying disjunctive database. By considering the different complete reduction orders we obtain different sets of clauses, each one representing some subset of the set of

preferred models. This provides an effective computation of the preferred models of a strict ordered database.

In its actual form the process is inefficient because the reductions are done once the whole characteristic set of the underlying disjunctive database is calculated. The reductions should be done earlier in order to eliminate needless clauses (for example, if A must be eliminated from the clause $A \vee B$, the clause $B \vee C$ need not to be generated from the rule $A \rightarrow C$). One can easily prove that the reduction process "commutes" with the generation of characteristic clauses. Consequently one can associate with any complete reduction order r, a (non-monotonic) operator T_r generating a reduced set of clauses after some number of iterations. These operators T_r define the least fixed point semantics of the strict ordered databases.

Proposition 4.13 (Least fixed point semantics for ordered databases). *Let SODB* = (DDB, $<$) *be a strict ordered database. Let T be the characteristic operator of DDB* ($T = \mathcal{T}_{\text{DDB}}$, *cf. Section* 2). *For any complete reduction order r for SODB,*[11] *let* T_r *be the operator defined by*: $T_r = r\text{-reduct} \circ \mathcal{T}_{\text{DDB}}$. *Then there exists an integer N such that*:

- $T_r^N(\ )$ *is a least fixed point of* T_r,
- $T_r^N(\ ) = r\text{-reduct}(\text{Char}(\text{DDB}))$.

5. Comparison between the Perfect Model and the Making Choices semantics

One can associate two different preferential semantics for a given non-definite database NDB (that is set of rules $A_1 \ldots A_n . \neg B_1 \ldots \neg B_m \rightarrow C_1 \ldots C_p$).

- The Perfect Model semantics: NDB is seen as the couple (DDB, $\lessdot$) where DDB is the underlying disjunctive database of NDB and $\lessdot$ is the model *preferability* relation associated with the priority relation $<$ of NDB (in the sense of Section 3).
- The Making Choices semantics where NDB is seen as the ordered database (DDB, $<$) for a preference relation $<$ defined as follows: $A < B$ iff there is a ground instance of a rule of NDB, $A_1 \ldots A_n . \neg B_1 \ldots \neg B_m \rightarrow C_1 \ldots C_p$, such that A is equal to some B_i and B is equal to some C_j. In this case the preferential semantics is defined by the relation $\ll$ (in the sense of Section 4).

We want to compare these two preferential semantics for a given non-definite database NDB. The work consists of comparing the maximal elements of the relations $\lessdot$ and $\ll$, both defined on the models of the underlying disjunctive database of NDB.

From the definitions of Sections 3 and 4 it follows that the priority relation $<$ of NDB is exactly the "backward chaining" closure[12] of the relation $<_*$. Thus the model relations $\lessdot$ and $\ll$ are not exactly defined from the same atomic relation.

[11] It means r is a complete reduction order for Char(DDB) with respect to $<_*$.

[12] Closure by the law: if $B < C$ and there is a rule $\ldots A \ldots \rightarrow \ldots B \ldots$ then $A < C$.

However one proves the following result (we give a sketch of proof in the appendix):

If M and N are two minimal models of DDB such that $M \ll N$,
there is a minimal model N' of DDB such that $M \ll N'$.

Consequently *any preferred model of the ordered database translation of non-definite database NDB is a perfect model of NDB.* The converse is false as shown in the example below. However, for stratified logic programs, the two semantics are equivalent because there is a unique perfect model.

Example 5.1. NDB $= \{A \vee B, \neg B \rightarrow C\}$. The perfect models of NDB are $\{A, C\}$ and $\{B\}$. The ordered database translation of NDB, ODB $= \{A \vee B;\ B \vee C\}$ with $B < C$, admits only $\{A, C\}$ as preferred model.

This example shows that the notion of preference involved in the Making Choices semantics is stronger than that related to Negation by Failure in Perfect Model semantics. The difference is that the Making Choices selection of models forces the derivation of preferred atoms. Another difference is that the elimination of a less preferred atom depends on the presence of a higher preference atom in the database, which is a notion quite different from the Negation as Failure principle.

This particular feature makes the operational semantics of Making Choices less complex than the operational semantics of Perfect Model semantics presented in Section 3. The reduction process is less expensive than the computation of residues in Subsection 3.2. In this sense the Making Choices semantics is certainly more tractable than Perfect Model semantics.

6. Conclusion

We have proposed a formalism to handle incomplete information in deductive databases. The idea was to formalize the control (or reduction) of incompleteness as an expression of some preferences among the models of the underlying database. We have illustrated the approach by considering two types of "preferential semantics":

- the well-known Perfect Model semantics for non-definite databases which captures the Negation as Failure rule;
- the Making Choices semantics which is a new type of preferential semantics formalizing a notion of making choices in disjunctive databases.

Our main contribution concerns the computational aspects. For the two preferential logics studied here we have defined effective algorithms, allowing the computation of the intended models under some simple calculability conditions (in both cases a non-cyclicity condition of the atomic preference relation). The Making Choices semantics presented here supports a reasonable least fixed point semantics. This gives some confidence in the practical use of the concept of ordered databases.

However the least fixed point semantics presented in this paper are still far from effective query evaluation algorithms. In the future we plan to study this problem.

For example, one can ask how the backward chaining strategies, as developed for example in the Alexander or General Magic Sets Methods [2, 26] for definite databases, could be extended to the non-definite or ordered databases. The challenge is how the backward chaining principle, the fundamental evaluation strategy in Logic Programming, can be adapted to the treatment of incomplete (disjunctive) information and how it can accommodate the selection of preferred atoms.

One motivation for having defined the operational semantics in this paper as least fixed point semantics lies in the hope that they will be well adapted for query evaluation. The interest of a least fixed point semantics lies indeed in the representation of the intended models by clauses. In this setting, the problem of an efficient query evaluation could be reduced to the computation by a backward chaining strategy of a set of clauses pertinent for the query (in a sense specific to each preferential semantics).

The problem is surely not an easy one. The complexity and untractability of the algorithms proposed in the literature for "computing" the non-monotonic logics, for example MILO-resolution [24], shows that the problem is inherently difficult and still needs much study. We think that the least fixed semantics approach, taking into account the particular nature of deductive databases, is the right way to attack the problem.

Besides the computational aspects, we also want to emphasize that the treatment of incomplete information as an expression of preferences can help in a methodical specification of databases. A main advantage of the preferential formalism is the clear separation between the database specification and the control of incomplete information, that is the definition of the preference relation. The gain is a greater modularity and flexibility. This also allows us to escape from the syntax of the rules, by identifying the essential specifications of the database.

For example, let us consider the following logic program

$$P = \{\neg A \to B, \neg C \to A, \neg B.\neg A \to C\}.$$

The program is not locally stratified because there is a cycle in the negative dependencies: $A < B < C < A$. The program is not even effectively stratified [5]. Thus there is no perfect model. However the program has an intuitive meaning, though not captured by Perfect Model semantics. Following the default logic approach of [4], the program should be interpreted as the default theory $\{: \neg A/B, : \neg C/A, : \neg B \wedge \neg A/C\}$ which has a unique extension $\{A\}$ corresponding to the intuitive semantics of P.[13]

It is worth examining the program with respect to the preference paradigm and to answer the two questions: what is the underlying incomplete database and what are the preferences expressed about the atoms? It is found that the underlying database is $\mathrm{DDB} = \{A \vee B, C \vee A, B \vee A \vee C\}$. The third clause is subsumed by the two others, so the database is in fact $\{A \vee B, C \vee A\}$. Consequently only the two first

[13] The third rule is not applicable because it is not compatible with the first one; the first one cannot be applied otherwise the second one will produce an inconsistency.

rules are significant and may define significant preference choices. Then the preferences induced by the two first rules are: $A < B$ and $C < A$. For both semantics, Perfect Model and Making Choices, the selected model is $\{A\}$. Hence the discipline imposed by the preferential formalism has helped to discover the right way to define the semantics of the program.

Appendix A: proofs

A.1. The preferred models are perfect models for stratified databases

Let SDB be a locally stratified database. Let $<$ the priority relation for BDS. Let ODB = (DDB, $<$) be the ordered database associated with SDB, that is DDB is the disjunctive database associated with SDB and $<$ is the set of couples $(B.A)$ such that there is an instance of rule in SDB of the form: $\ldots \neg B \ldots \rightarrow \ldots C \ldots$.

It follows by the definitions that $<$ is the forward and backward closure of $\prec$ (cf. Section 2).

Let $\lessdot$ be the preferability relation of SDB and $<_m$ the preference relation of its ordered database translation ODB. To show that any preferred models of ODB are perfect models of SDB, it is sufficient to prove the following property.

Property A.1. *For any two models M and N of DDB such that $M \neq N$ and $M \lessdot N$, there exists N' such that $M <_m N'$.*

Proof. We just give the essential ideas of the proof. Let M and N be two models satisfying the hypothesis of Property A.1.

(1) If the only couples involved in the comparison $M \lessdot N$ are couples of $<_*$ then trivially we have $M <_m N$. Thus $N' = N$ in Property A.1.

(2) The problem comes from couples (A, C) involved in the comparison of $M \lessdot N$ which are obtained by backward chaining from $\prec$. N' will be constructed from N by elimination of these such couples in each stratum successively of SDB.

(3) Let n be the least integer such that the set of couples (A, C) of $M \setminus N \times N \setminus M$ with $A < C$ and level[14] $(C) = n$ is not empty. Let (A, C) be one particular couple of this set. The case where $A < C$ cannot be used as a justification for C in the sense of $<_m$ is the case where $C <_* G$ for some G in $M \setminus N$ and for no atom D of $M \setminus N$, $D <_* C$. Then the typical situation is the following:

- There exist an atom B and two instances of SDB rules of the form $U, \ldots, \neg B \ldots \rightarrow C \ldots$ and $A, \ldots \rightarrow B, \ldots$.
- A does not belong to N, hence it does not appear as unit consequence of some DDB rule. Thus N contains an atom F (not in M) of the same level ($<n$) of A. Let $F1$ be the justification of F in M w.r.t. $\lessdot$: by the hypothesis on n, we have $F1 <_* F$, hence $F1$ is a justification of F w.r.t. $<_m$.

[14] Where level is the stratification function chosen for SDB.

- B belongs to neither M nor N.
- C being in N, probably U belongs to N but it is surely not in M. If U is in N, it needs a justification: as it is of level $<n$ we can suppose that there is a rule instance of the form $\ldots \neg V \rightarrow U \ldots$ with V in $M \setminus N$.[15]
- $G \in M$ and $C <_* G$. Again we suppose that there is a rule instance of the form $\ldots \neg C \rightarrow G \ldots$.

Thus we obtain $M = \{A, F1, G, \ldots\}$ and $N = \{F, U, C \ldots\}$ with some rules: $\ldots \rightarrow A \vee F \ldots$, $\ldots \neg F1 \rightarrow F \ldots$, $\ldots A \rightarrow B \ldots$, $\ldots \neg B \rightarrow C \ldots$, $\ldots \neg V \rightarrow U \ldots$ and $\ldots \neg C \rightarrow G \ldots$. We do not yet have $M <_m N$ because C lacks a justification $C1$ in $M \setminus N$ such that $C1 <_* C$. Thus we suppress C from N and instead define N' as follows: $N' = \{F, V, G, \ldots\}$. Formally N' coincides with N for the atoms of level $<n$, except for U which is replaced by V. If this process is repeated for every problematic couple (A, C) we finally get models M and N' such that their restrictions M_n and N_n to the nth stratum verify $M_n <_m N'_n$. □

A.2. Proof of the validity of the reduction process

Let ODB = (DB, $<$) be an ordered database. Let r be a reduction order compatible with $<_*$ and R be the set of clauses resulting from the r-reduction of Char(DB). We also suppose that R is reduced w.r.t. $<$.

(a) *Minimal models of R are minimal models of DB*: It is easy to prove that the minimal models of R are minimal models of the original set of clauses Char(DB). It is sufficient to show that this property is true for the reduction of a single atom A.

- Let M be a model of R the A-reduction of Char(DB). Then M is clearly a model of the subset of clauses in Char(DB) containing A. For a clause of Char(DB) not containing A, either it still belongs to R or it is eliminated by a reduced clause of Char(DB). In both cases it is true in M.
- Now let us consider $M1 = M \setminus B$ for some atom B of M. M being minimal for R, $M1$ invalidates some clause C of R. Either C belongs to Char(DB), hence $M1$ is not model of Char(DB), or C comes from the A-reduction of a clause D of Char(DB). In this latter case, as A does not belong to M, $M1$ invalidates D, hence is not model of Char(DB). Consequently M is minimal for Char(DB) also.

(b) *Models of R are maximal for* $<_m$: The proof proceeds by induction on the cardinal of $<$. We need the following lemma.

Lemma A.2. *Let M be a minimal model of DB and A an atom of DB not in M. Then M is a minimal model of the A-reduction of* Char(DB), *the set of characteristic clauses of DB.*

Proof. M is a minimal model of Char(DB). As A is not in M, the elimination of A in the clauses of Char(DB) yields clauses which are still true in M. By minimality of M, $M1 = M \setminus \{B\}$ (for any B in M) invalidates some clause C of Char(DB). We

[15] The reasoning is more complicated but similar if the justification of U is obtained by $V <_* U$.

must prove that $M1$ invalidates some clause C' of R, the A-reduction of Char(DB). If C belongs to R, we can take $C' = C$. If A belongs to C, we can take $C' =$ the A-reduction of C (because $A \notin M$). If C is eliminated from Char(DB), it is necessary because it is subsumed by a clause D of R coming from a clause $A \vee D$ of Char(DB). In this latter case we can take $C' = D$. □

Induction base: If the preference relation is empty there is almost nothing to prove. The model preference relation is also empty and every model is trivially maximal. On the other hand the r-reduction of Char(DB) is Char(DB) itself.

Induction step: If there is no atom eliminated in the reduction process this means that all the reducible atoms appear as unit clauses in Char(DB) hence they all belong to every model of DB. Thus the models are uncomparable with each other (strictness condition fails each time) and every model is maximal.

Let us suppose now that A is the first atom to be (effectively) eliminated in the r-reduction of Char(DB). A does not belong to M. In a comparison $M <_m M'$, where M' is a model of Char(DB), it is impossible that A belongs to M' (else A would have no justification in M). Hence if $M <_m M'$ then, by Lemma A.2, M' is a model of $R1$ the A-reduction of Char(DB) and the comparison does not involve couples of the form $(A, \ldots)$. Thus we have $M <^1_m M'$ as models of $R1$, where $<^1_m$ is the model preference relation defined on $R1$ from the restriction $<^1$ of $<$ to the atoms of $R1$. The cardinal of $<^1$ being strictly less than the cardinal of $<$, we infer by induction hypothesis that M is maximal for $<^1_m$, which contradicts $M <^1_m M'$. Consequently M is maximal for $<_m$.

A.3. Proof of the completeness of the reduction process

Induction base: If the preference relation is empty the proof is again trivial. Every model of DB is preferred and every reduction order is empty thus any r-reduction of Char(DB) is Char(DB).

Induction step: We first remark that there is at least one minimal atom of $<$ not in M.

Lemma A.3. *If M is a preferred model of* (DB, $<$) *then M does not contain at least one minimal atom of $<$.*

Proof. Let $A1, A2, \ldots, An$ be the atoms of DB minimal for $<$. Let us suppose that M contains $A1, A2, \ldots, An$. As $A1 < B$ for some B which, by the hypothesis made on $<$, is not strongly dependent on $A1$, there is a minimal model M' of DB which contains B but not $A1$. Every ordered atom of $M' \setminus M$ has a justification in M (it is justified by some minimal element of $<$), hence $M < M'$, which contradicts the maximality of M. □

Let A be some minimal element of $\prec$ not belonging to M. Let $R1$ be the A-reduction of Char(DB). By Lemma A.2, M is a minimal model of $R1$. We claim that *M is also maximal for $\prec_m^1$ in the set of models of $R1$*, where $\prec_m^1$ is the model preference relation defined from the restriction $\prec^1$ of $\prec$ to the atoms of $R1$.

- This will be the case if we can prove that $\prec_m^1$ is the restriction of $\prec_m$ to the subset of models of $R1$. It is sufficient to prove that $(\prec^1)_*$ is the restriction of $\prec_*$ to the atoms of $R1$.
- To be rigorous we need to formally establish the link between the A-reduction of the set of characteristic clauses of DB and a similar notion of A-reduction directly defined on the facts and rules of DB. Indeed $R1$ is the set of characteristic clauses of the disjunctive database DB1 obtained from DB by removing A from the facts of DB and removing the (instances of) rules of DB containing A as premise. (We skip this proof.) Consequently the closure operations (forward chaining and propagation) commute with the restriction to $R1$ (or equivalently to DB1): $(\prec_*)|R1 = (\prec_*)|\mathrm{DB1} = (\prec|\mathrm{DB1})_*$.

Hence the induction hypothesis applies to M. There is a reduction order $r1$ compatible with $\prec_*^1$, complete with respect to $\prec^1$ and such that M is a minimal model of the $r1$-reduction of $R1$. To finish the proof, we show that $r1$ can be extended to a complete reduction order r compatible with $\prec_*$ and show that the $r1$-reduction of $R1$ is exactly the r-reduction of Char(DB) and thus characterizes the model M.

- r is defined by concatenating A to $r1$.
- r is trivially compatible with $\prec_*$, because we have added a minimal element of *prec* and $r1$ was compatible with $\prec_*^1$ which is a restriction of $\prec_*$.
- By definition of the reduction of a set of clauses, the $A.r$-reduction of a set of clauses E is equal to the r-reduction of the A-reduction of E. Thus the $r1$-reduction of $R1$ is equal to R, the r-reduction of Char(DB).
- r is complete with respect to $\prec$: if there subsists in R an atom B reducible with respect to $\prec$, B cannot be A; hence B belongs to $R1$ and B is necessarily reducible with respect to $\prec^1$ which contradicts the fact that $r1$ is complete with respect to $\prec^1$.

References

[1] K. Apt, H. Blair and A. Walker, Towards a theory of declarative knowledge, in: *Proc. Workshop on Foundations of Deductive Databases* (1986).

[2] F. Bancilhon and D. Maier, Magic sets and other strange ways to implement logic programs, in: *Proc. Symp. on Principles of Database Systems* (1986).

[3] P. Besnard and P. Siegel, The preferential-models approach to non-monotonic logics, in: *Non-Standard Logics for Automated Reasoning* (Academic Press, New York, 1988) 135–161.

[4] N. Bidoit and C. Froidevaux, Minimalism subsumes default logic and circumscription, in: *Proc. Logic in Computer Science* (IEEE, New York, 1987).

[5] N. Bidoit and C. Froidevaux, More on stratified default theories, in: *Proc. Europ. Conf. on Artificial Intelligence*, Munich (1988).

[6] G. Bossu and P. Siegel, Saturation, non-monotonic reasoning and the closed world assumption, *Artificial Intelligence* **25**(1) (1985).
[7] A. Brown and Y. Shoham, New results on semantical nonmonotonic reasoning, in: *Proc. 2nd Workshop on Non-Monotonic Reasoning*, Grassau, Lecture Notes in Artificial Intelligence **346** (Springer, Berlin, 1988) 19-41.
[8] C.L. Chang and R. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Computer Science and Applied Mathematics (Academic Press, New York, 1973).
[9] K.L. Clark, *Negation as Failure* (Plenum, New York, 1978) 293-322.
[10] R. Demolombe, An efficient strategy for non-Horn deductive databases, in: *Proc. IFIP*, San Francisco (1989).
[11] M. Gelfond, H. Przymusinska and T. Przymusinski, On the relationship between circumscription and negation as failure, *Artificial Intelligence* **38** (1989) 75-94.
[12] J.M. Kerisit, La Méthode Alexandre: une Méthode de Déduction, Thèse Université Paris 7, 1988.
[13] R. Kowalski and M.H. Van Emden, The semantics of predicate logic as a programming language, *J. ACM* **23**(4) (1976).
[14] V. Lifschitz, Closed world databases and circumscription, *Artificial Intelligence* **27** (1985).
[15] V. Lifschitz, Circumscriptive theories: a logic-based framework for knowledge representation, in: *Proc. AAAI* (1987) vol. 1.
[16] J. MacCarthy, Circumscription: a form of non-monotonic reasoning, *Artificial Intelligence* **13**(1, 2) (1980) 27-39.
[17] J. MacCarthy, Applications of circumscription to formalizing common-sense knowledge, *Artificial Intelligence* **28** (1986).
[18] D. Makinson, General theory of cumulative inference, in: *Proc. 2nd Workshop on Non-Monotonic Reasoning*, Grassau, Lecture Notes in Artificial Intelligence **346** (Springer Verlag, Berlin, 1988) 1-18.
[19] J. Minker, On indefinite databases and the closed world assumption, in: *Proc. 6th Conf. on Automated Deduction* (1982).
[20] J. Minker and A. Rajasekar, A fixpoint semantics for non-Horn logic programs, Technical Report UMIACS-TR-87-24, University of Maryland, IACS, 1987.
[21] J. Minker and A. Rajasekar, Procedural interpretation of non-Horn logic programs, in: *Proc. Conf. on Automated Deduction* (1988).
[22] T. Przymusinski, On the declarative semantics of stratified deductive databases, in: *Proc. Foundations of Deductive Databases and Logic Programming*, Washington (1986).
[23] T. Przymusinski, On the declarative and procedural semantics of logic programs, *J. Automat. Reason.* **5** (1989) 167-205.
[24] T. Przymusinski, An algorithm to compute circumscription, *Artificial Intelligence* **38**(1) (1989) 49-74.
[25] R. Reiter, A logic for default reasoning, *Artificial Intelligence* **13**(1, 2) (1980).
[26] J. Rohmer, R. Lescoeur and J.M. Kerisit, The Alexander Method: a technique for the processing of recursive axioms in deductive databases, *New Generation Comput.* **3**(4) (1986).
[27] V. Royer, Modeling preference choices in incomplete deductive databases, in: *Proc. IFIP*, San Francisco (1989).
[28] V. Royer, Backward chaining evaluation in stratified disjunctive databases, Technical Report Esprit Project ESTEAM 316, final deliverable, Oct. 1989.
[29] Y. Shoham, Nonmonotonic logics: meaning and utility, in: *IJCAI 10*, Milan (1987).
[30] Y. Shoham, *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence* (MIT Press, Cambridge, MA, 1988).
[31] J.E. Stoy, *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory* (MIT Press, Cambridge, MA, 1977).