Towards an Operational Semantics of Rules in Knowledge Systems

Mohamed A. Khamsi* Driss Misane† Gerd Wagner‡

Abstract

We develop a general closure semantics for deduction rules in knowledge bases, using the concept of a *(deductive) knowledge system* proposed in [Wag94a] where rules are interpreted as update functions operating on knowledge bases. We first present four important examples of basic knowledge systems: relational databases, (defeasible) factbases, temporal databases, and epistemic states. We then define the notion of a *supported closure*, and we show that every ampliative deductive knowledge base has a supported closure. Choosing those supported closures which satisfy a certain stability condition as the preferred (or intended) ones we obtain the *stable closure semantics* for deductive knowledge bases (including normal, extended, disjunctive, and other logic programming and rule-based systems).

Contents

*Department of Mathematics, Univ. of Texas at El Paso, TX 79968, E-Mail: mohamed@math.ep.utexas.edu
†Univ. Mohammed V, Faculte des Sciences, Dep. de Mathematiques, Rabat, Moroco
‡Univ. Leipzig, Inst.f.Informatik, Augustusplatz 10-11, D-04109 Leipzig, Germany, gw@inf.fu-berlin.de

1 Introduction

While reasoning on the basis of rule knowledge has turned out to be essential in AI, standard logics do not allow for rules as expressions of the representation language. The only possibility to process rule knowledge in a standard logic is to translate a rule expression $F \leftarrow G$ into an implicational formula $G \rightarrow F$ of the resp. language.[1] Such a translation, however, is only faithful in special cases. For instance, in the case of positive logic programming rules $a_0 \leftarrow a_1 \wedge \ldots \wedge a_n$, it is possible to process them as implications $a_1 \wedge \ldots \wedge a_n \rightarrow a_0$ in classical or intuitionistic logic, or even as definite clauses $\neg a_1 \vee \ldots \vee \neg a_n \vee a_0$ in classical or three-valued logic, without making any difference. This is no longer the case when the rule language is more expressive, i.e. when the conclusion or the premise of a rule contain additional logical operators, such as disjunction, or negation.

Historically, this difficulty is reflected in the recent shift in logic programming semantics, abandonning earlier classical logic approaches, such as the Horn clause reading for positive logic programs, or Clark's completion semantics for positive and normal logic programs, and seeking for new rule-oriented semantics, such as the answer set semantics of [GL90], or the partial semantics of [Wag91], for extended logic programs. In the latter proposals, and in the proposal to be presented below, the semantics of rules does not imply the Contraposition principle, which would allow to infer $\sim q$ from $\{p, \sim p \leftarrow q\}$. Contraposition, however, is essential in most standard semantics of implication, such as in classical, intuitionistic, or relevance logic. A notable exception where Contraposition does not hold is Nelson's *constructive logic with strong negation* $\mathbf{N}$, and indeed, as shown in [Pea93], a rule of an extended logic program can be interpreted as a constructive implication in $\mathbf{N}$. However, rules operating on disjunctive information, such as in

$$\{p \vee q,\ r \leftarrow p,\ r \leftarrow q\}$$

differ from constructive implication: in $\mathbf{N}$, we would obtain r (the corresponding inference principle is called 'Disjunction in the Premise'), whereas a rule-oriented semantics would not necessarily allow this inference.

In this paper, we present a new and direct approach to the semantics of deduction rules.[2] The key concepts for this approach are that of a *knowledge system*, and that of a *deductive knowledge*

[1]Another possibility, in the spirit of Gödel's translation $*$ of intuitionistic logic into $\mathbf{S4}$, is the explicit epistemic interpretation of a rule $F \leftarrow G$ as $\Box(G^* \rightarrow F^*)$.

[2]We do not identify 'semantics' with model theory. There are also proof-theoretic semantics of logical systems. Only in 'nice' cases, e.g. in classical logic, it is clear that model theory is more fundamental (conceptually simpler and more 'declarative') than proof theory. There are other cases, however, where this is not clear (e.g. relevance

base (DKB), proposed in [Wag94a]. Inspired by Belnap's notion of an *information state* [Bel77], rules are interpreted as update functions operating on knowledge bases, and the semantics of a deductive knowledge base is determined by the existence of preferred closures being common fixpoints of all rules. Our theory, therefore, is both a fixpoint theory for certain nonmonotonic functions, and a semantical theory for the concept of *deductive knowledge bases* comprising various deductive database and logic programming systems. It is conceptually simpler than many other (quite technical) proposals on the semantics of logic programming rules, and it is at the same time more general.

While monotonic ampliative DKBs have a unique minimal closure which is naturally the intended one, one has to find an appropriate preference criterion in order to define the intended closures of a nonmonotonic DKB. Such a preference criterion then selects from the supported closures the intended ones in a conservative fashion: in the special case of a monotonic ampliative DKB this has to be the unique minimal closure.

Since nonmonotonic rules do not necessarily contain negation-as-failure (there are other non-persistent operators, such as exclusive disjunction, or certain modal operators), a general semantics for DKBs cannot refer to negation-as-failure, or any other specific logical operator. Rather, it has to account for the way in which the application (successive detachment) of a set of rules yields an intended closure. It turns out that the notion of a *stable closure* is a key for the semantics of DKBs. While certain 'well-behaved' DKBs have a unique stable closure, there may be several stable closures, or none, in the general case.[3]

2 Basic Concepts

A *signature* $\sigma = \langle Rel, Fun, Const \rangle$ consists of a set of relation symbols *Rel*, a set *Fun* of function symbols, and a set of constant symbols *Const*. We consider the following logical functors: conjunction ($\wedge$), disjunction ($\vee$), strong negation ($\sim$), weak negation (alias negation-as-failure, denoted by $-$), exclusive disjunction ($|$), and the truth constant 1; relation symbols are denoted by $p, q, r, \ldots$; constant symbols by $c, d, \ldots$; and variables by $x, y, \ldots$. Quantifiers, $\exists$ and $\forall$, are only incidentally considered. If $\mathcal{F}$ is a set of logical functors, $L(\sigma; \mathcal{F})$ denotes the corresponding set of wellformed formulas. $L(\sigma) = L(\sigma; -, \sim, \wedge, |, \vee)$ is the smallest set containing the atomic formulas of σ, and being closed with respect to the following condition: if $F, G \in L(\sigma)$, then $\{-F, \sim F, F \wedge G, F \vee G, F|G\} \subseteq L(\sigma)$.

With respect to a signature σ we define the following sublanguages: $\mathrm{At}(\sigma) = L(\sigma; \emptyset)$, the set of all atomic sentences (also called *atoms*); $\mathrm{Lit}(\sigma) = L(\sigma; \sim)$, the set of all *literals*; and $\mathrm{XLit}(\sigma) = \mathrm{Lit}(\sigma) \cup \{-l : l \in \mathrm{Lit}(\sigma)\}$, the set of all *extended literals*. We shall frequently omit the reference to a specific signature, and simply write L instead of $L(\sigma)$. We introduce the following convention: when L is a set of sentences, L^x denotes the corresponding set of formulas.

logics, or default logic). Our definition of a knowledge system does not require a model-theoretic semantics, but neither does it preclude one.

[3]We shall present a generalization of the logic programming concept of *stratification* for deductive knowledge bases in a sequel paper where we show that, mathematically, the notion of stratification is not related to negation-as-failure, but rather to the problem of the computational stability of a set of nonmonotonic rules and the uniqueness of the resp. closure.

An *atom* $a \in \mathrm{At}$ is called *proper*, if $a \neq 1$. We use $a, b, \ldots,\ l, k, \ldots$, and $F, G, H, \ldots$ as metavariables for atoms, literals, and well-formed formulas, respectively.

With each negation a complement operation for the resp. type of literal is associated: $\tilde{a} = {\sim}a$ and $\widetilde{{\sim}a} = a$, $\bar{l} = -l$ and $\overline{-l} = l$. These complements are also defined for sets of resp. literals $L \subseteq \mathrm{Lit}$, and $E \subseteq \mathrm{XLit}$: $\widetilde{L} = \{\tilde{l} : l \in L\}$, resp. $\overline{E} = \{\bar{e} : e \in E\}$. We distinguish between the positive and negative elements of $E \subseteq \mathrm{XLit}$ by writing $E^+ := E \cap \mathrm{Lit}$ and $E^- := \{l : -l \in E\}$.

If Y is an ordered set, then $\mathrm{Min}(Y)$ denotes the set of all minimal elements of Y, i.e. $\mathrm{Min}(Y) = \{X \in Y \mid \neg\exists X' \in Y : X' < X\}$

2.1 Knowledge Systems

Before presenting the formal definitions, we start with a semi-formal discusssion of the basic concepts to be introduced, notably: knowledge base, query, inference, answer, information ordering, input and update.

In general, a knowledge base (KB) can consist of any kind of data structures capable of representing knowledge, e.g. a set, or multiset, or sequence, of (logical) expressions, or a directed graph, etc. For the sake of simplicity, we shall assume that a KB is a set of expressions from a representation language. Only certain formulas may make sense for representing knowledge, that is, there will be a specific representation language L_{Repr}, and a KB will be a (usually finite) collection of elements of L_{Repr}, possibly constrained in some way determined by the set L_{KB} of all admissible KBs: $\mathrm{KB} \in L_{\mathrm{KB}} \subseteq 2^{L_{\mathrm{Repr}}}$. Likewise, since not every formula may be appropriate as a sensible query, the set of admissible queries is specified by L_{Query}.

The basic scenario of a knowledge system (KS) consists of two operations: an inference operation processing queries posed to the KB, and an update operation processing inputs entered by users or by other (e.g. sensoric) information suppliers. A KS restricts the admissible inputs to elements of a specific input language L_{Input}, and an update is performed by processing the input formula in an appropriate way in order to assimilate its information content into the KB. Since it appears reasonable to require that any information entered to a KB can be queried afterwards, we shall assume that $L_{\mathrm{Input}} \subseteq L_{\mathrm{Query}}$.

Definition 1 (Knowledge System) *An abstract knowledge system $\boldsymbol{K}$ is a quintuple:*[4]

$$\boldsymbol{K} = \langle\, L_{\mathrm{KB}},\ \vdash,\ L_{\mathrm{Query}},\ \mathsf{Upd},\ L_{\mathrm{Input}}\,\rangle$$

where the inference relation $\vdash\ \subseteq L_{\mathrm{KB}} \times L_{\mathrm{Query}}$, together with the update operation $\mathsf{Upd} : L_{\mathrm{KB}} \times L_{\mathrm{Input}} \to L_{\mathrm{KB}}$, satisfy for any $X \in L_{\mathrm{KB}}$,

(KS1) $X \vdash 1$, *and* $\mathsf{Upd}(X, 1) = X$.

(KS2) $L_{\mathrm{Input}} \subseteq L_{\mathrm{Query}}$.

[4]The formulation of a KS in terms of query and input processing was already implicitly present in Belnap's [1977] view of a KS. In [Lev84] it was proposed as a 'functional approach to knowledge representation'. In [Wag94a, Wag95a] the concept of knowledge systems was further extended and used as an integrating framework for knowledge representation and logic programming.

(KS3) $\mathsf{Upd}(X,F) \vdash F$, *for any* $F \in L_{\text{Input}}$ *which is consistent with* X.[5]

If elements of L_{KB} are finite sets (resp. structures), $\boldsymbol{K}$ is called *finitary.* In the sequel, we shall sometimes simply write 'KB' in formal expressions standing for an arbitrary knowledge base $X \in L_{\text{KB}}$. An inference operation C is defined as usual:

$$C(\text{KB}) = \{F \in L_{\text{Query}} : \text{KB} \vdash F\}$$

In many cases, it is useful to be able to update by a set of inputs and we 'overload' the symbol Upd to denote also this more general update operation

$$\mathsf{Upd} : L_{\text{KB}} \times 2^{L_{\text{Input}}} \to L_{\text{KB}}$$

which has to be defined in such a way that for any finite $A \subseteq L_{\text{Input}}$, $\mathsf{Upd}(\text{KB}, A) = \mathsf{Upd}(\text{KB}, \bigwedge A)$. We sometimes write $\text{KB} + F$ as an abbreviation of $\mathsf{Upd}(\text{KB}, F)$, resp. $\text{KB} - F$ as an abbreviation of $\mathsf{Upd}(\text{KB}, -F)$. Similarly,

$$\text{KB} + \sum_{i=1}^{m} F_i = (\ldots((\text{KB} + F_1) + F_2) + \ldots) + F_m$$

Example 1 (Relational Databases) *A KB consisting of ground atoms corresponds to a relational database.*[6] *For instance,* $X = \{r(S), m(P,L)\}$ *may represent the information that Susan is a resident, and that Peter is married with Linda.*

As a kind of natural deduction from positive facts an inference relation $\vdash$ *between a relational database* $X \subseteq \text{At}$ *and a ground formula* $F \in L(-,\wedge,\vee,|)$ *is defined in the following way:*

$$\begin{array}{lrll} (\vdash a) & X \vdash a & \textit{if} & a \in X \\ (\vdash -a) & X \vdash -a & \textit{if} & a \notin X \\ (\vdash \wedge) & X \vdash F \wedge G & \textit{if} & X \vdash F \ \ \& \ \ X \vdash G \\ (\vdash \vee) & X \vdash F \vee G & \textit{if} & X \vdash F \ \textit{or}\ X \vdash G \\ (\vdash |) & X \vdash F|G & \textit{if} & X \vdash F \wedge -G \ \textit{or}\ X \vdash G \wedge -F \end{array}$$

This inductive definition is completed by assuming the following DeMorgan-style rewrite rules:

$$\begin{array}{rcl} -(F \vee G) & \longrightarrow & -F \wedge -G \\ -(F \wedge G) & \longrightarrow & -F \vee -G \\ --F & \longrightarrow & F \end{array}$$

Updates are insertions, $\mathsf{Upd}(X,a) := X \cup \{a\}$, *and deletions,* $\mathsf{Upd}(X,-a) := X - \{a\}$. *For consistent* $E \subseteq \text{At} \cup \overline{\text{At}}$, *we have* $\mathsf{Upd}(X,E) = X \cup E^{+} - E^{-}$. *The KS of relational databases, denoted by* $\boldsymbol{A}$, *is then defined as*

$$\boldsymbol{A} = \langle 2^{\text{At}}, \vdash, L(-,\wedge,\vee,|), \mathsf{Upd}, \text{At} \cup \overline{\text{At}} \rangle$$

If $L_{\text{Input}} = \text{At}$, *and in addition* $L_{\text{Query}} = L(\wedge,\vee)$, *the resulting system without weak negation will be denoted by* $\boldsymbol{A}^{+}$.

[5]According to some notion of consistency associated with the abstract knowledge system. E.g., one might want to exclude contradictory pieces of information from this reflexivity principle: $\mathsf{Upd}(\{\sim p\}, p) \not\vdash p$. We shall not discuss this issue in the present paper, however.

[6]Model-theoretically, a relational database corresponds to a finite first order interpretation.

2.2 Regular Knowledge Systems

In order to compare knowledge bases in terms of their information content we assume that there is an *information*, or *knowledge ordering* $\leq$ between KBs such that

$\mathrm{KB}_1 \leq \mathrm{KB}_2$ *if* KB_2 *contains at least as much information as* KB_1.

The information ordering should be defined in terms of the structural components of knowledge bases and not in terms of higher-level notions (like derivability).[7] The informationally *empty KB* will be denoted by 0. By definition, $0 \leq X$ for all $X \in L_{\mathrm{KB}}$, i.e. 0 is the least element of $\langle L_{\mathrm{KB}}, \leq \rangle$.

In general, more information does not mean more consequences. In other words: answers are not necessarily preserved under growth of information. Queries, for which this is the case, are called *persistent*.

Definition 2 (Persistent Queries) *A closed query formula F is called* persistent (anti-persistent) *if $\forall X_1, X_2 \in L_{\mathrm{KB}} : X_1 \vdash F$ implies $X_2 \vdash F$, whenever $X_1 \leq X_2$ ($X_1 \geq X_2$). If all $F \in L_{\mathrm{Query}}$ are persistent, the KS and its inference relation $\vdash$ are called persistent. The set of all persistent query formulas is denoted by L_{PersQ}. An operator of the query language is called persistent, if every query formed with it and with persistent subformulas is again persistent.*

Definition 3 (Ampliative Inputs) *An input formula F is called (i)* ampliative[8] *if* $\mathrm{KB} \leq \mathsf{Upd}(\mathrm{KB}, F)$, *or (ii)* reductive *if* $\mathrm{KB} \geq \mathsf{Upd}(\mathrm{KB}, F)$. *A KS and its update operation* Upd *are called* ampliative, *if all inputs $F \in L_{\mathrm{Input}}$ are ampliative. The set of all ampliative input formulas is denoted by L_{AmpI}.*

A certain subset $L_{\mathrm{Unit}} \subseteq L_{\mathrm{Input}}$ designates those elementary expressions which will be called *information units*, e.g. atoms, literals, or weighted (resp. labelled, or annotated) atoms, and the like. An information unit represents an elementary piece of information with a positive information content. A knowledge base may contain contradictory pieces of information, and we assume that all inconsistent information units contained in $X \in L_{\mathrm{KB}}$ are collected by $\mathrm{Inc}(X) \subseteq L_{\mathrm{Unit}}$.

Definition 4 (Regular KS) *A knowledge system $\boldsymbol{K}$ is called* regular, *if there is a preorder $\langle L_{\mathrm{KB}}, \leq, 0\rangle$ with least element 0, a designated set $L_{\mathrm{Unit}} \subseteq L_{\mathrm{Input}}$, and an operation $\mathrm{Inc} : L_{\mathrm{KB}} \to 2^{L_{\mathrm{Unit}}}$, such that*

(KS4) Unit inputs increase the information content (at least if they are consistent): $X \leq X + u$, for any $X \in L_{\mathrm{KB}}$, and for any $u \in L_{\mathrm{Unit}}$, such that $u \notin \mathrm{Inc}(X)$, and $\mathrm{Inc}(X+u) \subseteq \mathrm{Inc}(X)$.

(KS5) The information ordering is compatible with ampliative update and persistent inference: for all $X_1, X_2 \in L_{\mathrm{KB}}$,

$$X_1 \leq X_2 \quad \textit{iff} \quad \forall F \in L_{\mathrm{AmpI}} \forall G \in L_{\mathrm{PersQ}} : X_1 + F \vdash G \Rightarrow X_2 + F \vdash G$$

[7]The usual way to compare the information content of two KBs in standard logic, namely by means of checking the inclusion of consequences: $\mathrm{KB}_1 \leq \mathrm{KB}_2$ if $C(\mathrm{KB}_1) \subseteq C(\mathrm{KB}_2)$, does not work in a nonmonotonic setting.

[8]The name is adopted from [Bel77].

(KS6) Ampliative inputs are persistent queries: $L_{\mathrm{AmpI}} = L_{\mathrm{PersQ}} \cap L_{\mathrm{Input}}$.

(KS7) Consistent Inference: for any $X \in L_{\mathrm{KB}}$*, and any* $F \in L_{\mathrm{Input}}$*,* $X \vdash F$ *implies* $\mathrm{Inc}(X+F) \subseteq \mathrm{Inc}(X)$.

A regular KS will be represented as a 9-tuple

$$\langle\, 0,\ \leq,\ L_{\mathrm{KB}},\ \vdash,\ L_{\mathrm{Query}},\ \mathsf{Upd},\ L_{\mathrm{Input}},\ \mathrm{Inc},\ L_{\mathrm{Unit}}\,\rangle$$

Example 2 (Standard Logics) *A standard logic (such as classical, or intuitionistic, logic), given by a language* L *and a consequence relation* $\vdash \subseteq 2^L \times L$*, resp. by the corrresponding consequence operation* C*, can be viewed as an infinitary knowledge system*

$$\langle\, \emptyset,\ \subseteq,\ \{X \in 2^L : X = C(X)\},\ \vdash,\ L,\ \mathsf{Upd},\ L,\ \mathrm{Inc},\ L\,\rangle$$

where 1) a KB is a deductively closed set of formulas, 2) the knowledge ordering is set inclusion, 3) update by F *is the addition of* F *and subsequent closure, i.e.* $\mathsf{Upd}(X,F) = C(X \cup \{F\})$*,*[9] *4) the query, unit and input languages are all equal to* L*, and 5)* $\mathrm{Inc}(X) = \emptyset$ *if* $X \neq L$*, and* $\mathrm{Inc}(X) = X$ *otherwise. KS1–KS7 hold, more or less, trivially. Notice, however, that it is not clear whether a standard logic corresponds to a sensible finitary knowledge system, because set inclusion is no longer an adequate knowledge ordering if KBs are not deductively closed (KS5 is violated).*

2.3 Further Examples of Knowledge Systems

2.3.1 Factbases

A KB consisting of ground literals (viewed as positive and negative facts) is called a *factbase*.[10] For instance, the factbase

$$X_1 = \{r(S), r(P), s(S), {\sim} s(L), {\sim} s(T),\ m(P,L),\ m(T,S)\}$$

may represent the information that Susan and Peter are residents, Susan is a smoker, Linda and Tom are nonsmokers, Peter is married with Linda, and Tom is married with Susan.

As a kind of natural deduction from positive and negative facts an inference relation $\vdash$ between a factbase $X \subseteq \mathrm{Lit}$ and a sentence is defined in the following way:[11]

$(\vdash a)$	$X \vdash a$	if	$a \in X$
$(\vdash {\sim} a)$	$X \vdash {\sim} a$	if	${\sim} a \in X$
$(\vdash -l)$	$X \vdash -l$	if	$l \notin X$
$(\vdash \wedge)$	$X \vdash F \wedge G$	if	$X \vdash F$ & $X \vdash G$
$(\vdash \vee)$	$X \vdash F \vee G$	if	$X \vdash F$ or $X \vdash G$
$(\vdash \mid)$	$X \vdash F \mid G$	if	$X \vdash F \wedge -G$ or $X \vdash G \wedge -F$
$(\vdash \exists)$	$X \vdash \exists x F(x)$	if	$X \vdash F(c)$ for some constant c

[9] Notice that this corresponds to the AGM *expansion* of 'belief sets', see [Gär88].

[10] Model-theoretically, a factbase corresponds to a finite partial first order interpretation.

[11] This inductive definition is completed by DeMorgan-style rewrite rules including double negation rules such as ${\sim}{-}a \longrightarrow a$. See [Wag94a].

For instance, one might ask X_2 "Is someone married with a nonsmoking non-resident ?",

$$X_2 \vdash \exists xy : m(x,y) \wedge {\sim} s(y) \wedge -r(y)$$

As in $\boldsymbol{A}$, updates are insertions, $\mathsf{Upd}(X,l) := X \cup \{l\}$, and deletions, $\mathsf{Upd}(X,-l) := X - \{l\}$, but now of literals which are the information units of fact bases, $L_{\mathrm{Unit}} = \mathrm{Lit}$. For consistent $E \subseteq \mathrm{XLit}$, we have $\mathsf{Upd}(X,E) = X \cup E^+ - E^-$. The knowledge system of factbases is then defined as

$$\boldsymbol{F} := \langle \emptyset, \subseteq, 2^{\mathrm{Lit}}, \vdash, L(-,\sim,\wedge,\vee,|,\exists), \mathsf{Upd}, \mathrm{XLit}, \mathrm{Inc}, \mathrm{Lit} \rangle$$

where $\mathrm{Inc}(X) = X \cap \widetilde{X}$.

We have to show that KS1–KS7 hold. Proof: it is obvious that KS1–KS4 hold. Since $L_{\mathrm{AmpI}} = \mathrm{Lit}$, and $L_{\mathrm{PersQ}} = L(\sim,\wedge,\vee)$, KS5 follows by straightforward induction on the complexity of query formulas. KS6 and KS7 are again obvious. □

2.3.2 Defeasible Factbases

In a defeasible factbase, contradictory items invalidate (i.e. neutralize) each other. As a kind of defeasible deduction from positive and negative facts an inference relation $\vdash$ between a factbase $X \subseteq \mathrm{Lit}$ and a ground formula is defined in the following way:

$$\begin{array}{lrl} (\vdash l) & X \vdash l & \text{if} \quad l \in X \ \ \& \ \ \tilde{l} \notin X \\ (\vdash -l) & X \vdash -l & \text{if} \quad l \notin X \\ (\vdash \wedge) & X \vdash F \wedge G & \text{if} \quad X \vdash F \ \ \& \ \ X \vdash G \end{array}$$

Updates are insertions (in the sense of consolidating revision), and deletions of literals $l \in \mathrm{Lit}$:

$$\begin{array}{rcl} \mathsf{Upd}(X,l) & := & \begin{cases} X - \{\tilde{l}\} \cup \{l\}, \text{ if } l \in \mathrm{Inc}(X) \\ X \cup l, \text{ otherwise} \end{cases} \\ \mathsf{Upd}(X,-l) & := & X - \{l\} \end{array}$$

We also add a unary operator standing for recency-preferring revision:

$$\begin{array}{rcl} \mathsf{Upd}(X,*l) & := & X - \{\tilde{l}\} \cup \{l\} \\ X \vdash *l & \text{if} & X \vdash l \end{array}$$

Information growth in defeasible factbases may come about in two ways: by the expansion of the consistent information, or by the reduction of the inconsistent information represeneted. Thus, the knowledge ordering between defeasible factbases is defined by

$$X_1 \leq X_2 \quad \text{iff} \quad X_1 - \mathrm{Inc}(X_1) \subseteq X_2 - \mathrm{Inc}(X_2) \quad \& \quad \text{f.a. } l \in \mathrm{Inc}(X_1) : l \in X_2 \ \text{ or } \ \tilde{l} \in X_2$$

i.e. an inconsistent piece of information is considered as more informative than no information. For instance,

$$\{p\} < \{p, q, {\sim} q\} < \{p, q\}$$

The knowledge system of defeasible factbases is then defined as

$$\boldsymbol{F}_d := \langle \emptyset, \leq, 2^{\mathrm{Lit}}, \vdash, L(*,-,\sim,\wedge), \mathsf{Upd}, \mathrm{Lit}^{*,-} \mathrm{Inc}, \mathrm{Lit} \rangle$$

$\boldsymbol{F}_d$ is not regular since it violates KS6: there is no context-free class of ampliative inputs, i.e. $L_{\mathrm{AmpI}} = \emptyset$, while $L_{\mathrm{PersQ}} \cap L_{\mathrm{Input}} = \mathrm{Lit}$.

2.3.3 Temporal Databases

A temporal database is a set of timestamped atoms[12] of the form $a@T$, which we also consider as pairs $\langle a, T\rangle$, where $a \in \mathrm{At}$, and the timestamp T, representing *valid time*,[13] is a consistent list of closed intervals from a linear discrete temporal domain $\boldsymbol{T}$ (e.g. calendar dates, or the natural numbers), i.e. it has the form

$$[(b_1, e_1),\ (b_2, e_2),\ \ldots,\ (b_m, e_m)]$$

such that $b_i, e_i \in \boldsymbol{T}$, $b_i \le e_i$, and $e_i < b_{i+1}$. The set of all such timestamps is denoted by $\mathcal{T}$. We shall abbreviate the single interval list $[(b,e)]$ by $[b,e]$, and the single time point list $[(d,d)]$ by $[d]$ or just d.

The basic input of a temporal database is a timestamped atom $a@T \in \mathrm{At} \times \mathcal{T}$:

$$\mathsf{Upd}(X, a@T) = \begin{cases} X \cup a@T & \text{if there is no } S \in \mathcal{T} \text{ s.th. } a@S \in X \\ X - \{a@S\} \cup \{a@S+T\}; & \text{otherwise} \end{cases}$$

where $S + T$ is an appropriately defined merge operation for timestamps.

A temporal database can be queried whether a sentence holds or does not hold during a specific time period:

$$\begin{array}{rll} X \vdash a@T & \text{iff} & a@S \in X \ \&\ T \subseteq S \\ X \vdash -a@T & \text{iff} & a@S \notin X \text{ or } T \cap S = \emptyset \\ X \vdash (F \wedge G)@T & \text{iff} & X \vdash F@T \ \&\ X \vdash G@T \\ X \vdash (F \vee G)@T & \text{iff} & \text{f.a. } t \in T : X \vdash F@t \text{ or } X \vdash G@t \\ X \vdash (\exists x F(x))@T & \text{iff} & \text{f.a. } t \in T \text{ there is a constant } c \text{ s.th. } X \vdash F(c)@t \\ X \vdash \exists u F@u & \text{iff} & X \vdash F@t \text{ for some } t \in \boldsymbol{T} \end{array}$$

Non-timestamped queries are decided as follows:

$$X \vdash F \quad \text{iff} \quad X \vdash F@\mathbf{now}$$

In the sequel, we use $t, t_1, t_2, \ldots$ and $T, T_1, T_2, \ldots$ also for timepoint and timestamp variables if no confusion can arise. A complex temporal query, for instance, is "Are there persons who married the same person again ?", formally expressed as

$$\exists t_1, t_2, t_3 \exists x, y : t_1 < t_2 < t_3 \wedge m(x,y)@t_1 \wedge -m(x,y)@t_2 \wedge m(x,y)@t_3$$

The knowledge ordering between two temporal databases X_1 and X_2 is defined as follows. $X_1 \le X_2$ iff for all facts $a@S \in X_1$ there is a corresponding fact $a@T \in X_2$ such that $S \subseteq T$.

In summary, we obtain the following knowledge system:

$$T\boldsymbol{A} := \langle \emptyset,\ \le,\ 2^{\mathrm{At}\times\mathcal{T}},\ \vdash,\ L_{\mathcal{T}}(-,\wedge,\vee,\exists),\ \mathsf{Upd},\ \mathrm{At}\times\mathcal{T},\ \emptyset,\ \mathrm{At}\times\boldsymbol{T}\,\rangle$$

where $L_{\mathcal{T}} = L(-,\wedge,\vee,\exists) \times \mathcal{T}$.

[12]Model-theoretically, a temporal database corresponds to a temporal interpretation over a two-sorted first-order language.

[13]A more elaborate model of a temporal database would in addition include *belief time* as a second timestamp.

2.3.4 Epistemic States

An *epistemic state*[14] is a collection of sets of literals (each one describing a *possible situation*). For instance, if we know that Susan is not blonde and that Peter likes either Linda or Susan, we get the following KB:

$$Y_1 = \{\{\sim b(S),\, l(P,L)\},\, \{\sim b(S), l(P,S)\}\}$$

Formally, the system of epistemic states, denoted by $\boldsymbol{B}$, is defined as:

$$\boldsymbol{B} \;=\; \langle \{\emptyset\},\, \leq,\, 2^{2^{\text{Lit}}},\, \vdash,\, L(\sim,\wedge,\vee),\, \mathsf{Upd}_B,\, L(\sim,\wedge,\vee),\, \text{Inc}_B,\, \text{Lit}^{\vee}\,\rangle$$

where the information ordering between $Y_1, Y_2 \subseteq 2^{\text{Lit}}$ is defined as

$$Y_1 \leq Y_2 \quad :\Longleftrightarrow \quad \forall X_2 \in Y_2 \exists X_1 \in Y_1 : X_1 \subseteq X_2$$

yielding a preorder, and for $Y \subseteq 2^{\text{Lit}}$,

$$Y \vdash F :\Longleftrightarrow \text{for all } X \in Y : X \vdash F$$

where inference on the basis of literals, $X \vdash F$, is inference in $\boldsymbol{F}$. For instance, the query whether Peter likes someone who is either blonde or (definitely) not blonde is answered negatively:

$$Y_1 \not\vdash \exists x : l(P,x) \wedge (b(x) \vee \sim b(x))$$

Inputs are processed as follows

$$\begin{array}{lll}
(Ul) & \mathsf{Upd}_B(Y,l) & = & \{X \cup \{l\} : X \in Y\} \\
(U\vee) & \mathsf{Upd}_B(Y, F \vee G) & = & \mathsf{Upd}_B(Y,F) \cup \mathsf{Upd}_B(Y,G) \\
(U\wedge) & \mathsf{Upd}_B(Y, F \wedge G) & = & \mathsf{Upd}_B(\mathsf{Upd}_B(Y,F),G)
\end{array}$$

The elementary pieces of information in epistemic states are disjunctions of literals $l_1 \vee \ldots \vee l_m$. The set of all such disjunctions is denoted by $\text{Lit}^{\vee}$. The inconsistency measure Inc_B now collects all definite and indefinite contradictions:

$$\text{Inc}_B(Y) := \{\bigvee L : L \in \text{Min}(\{K \subseteq \text{Lit} \mid \forall X \in Y \exists l \in K : l \in X \cap \widetilde{X}\})\,\}$$

In order to define two other update operations differing from Upd_B with respect to inconsistency handling we first define two functions selecting from a set of possible situation descriptions the resp. acceptable ones:

$$\begin{array}{lll}
\text{Cons}(Y) & := & \{X \in Y \mid X \cap \widetilde{X} = \emptyset\} \\
\text{MInc}(Y) & := & \{X \in Y \mid \neg\exists Z \in Y : (Z \cap \widetilde{Z}) \subset (X \cap \widetilde{X})\}
\end{array}$$

The first operator, Cons, accepts only consistent situation descriptions, while the second one, MInc, accepts all situation descriptions which are minimally inconsistent. If an epistemic state

[14]Epistemic states were proposed by Belnap [1977] in order to establish a paraconsistent information processing system. The concept of a *disjunctive factbase*, introduced in [Wag93] and further investigated in [Wag95b], is a generalization of Belnap's epistemic states.

Y is consistent, i.e. if it contains at least one consistent epistemic alternative, then $\mathrm{Cons}(Y) = \mathrm{MInc}(Y)$. We can now define

$$\begin{aligned}\mathsf{Upd}_{ex}(Y,F) &:= \mathrm{Cons}(\mathsf{Upd}_B(Y,F))\\ \mathsf{Upd}_{mi}(Y,F) &:= \mathrm{MInc}(\mathsf{Upd}_B(Y,F))\end{aligned}$$

While Upd_B accepts inconsistent inputs in a liberally paraconsistent manner, Upd_{ex} does not accept inconsistent inputs at all. It implements the *ex contradictione sequitur quodlibet (ECSQ)* principle of classical logic by discarding all inconsistent epistemic alternatives. A good compromise between the hypersensitive inconsistency handling mechanism of Upd_{ex} and the too liberal Upd_B is the principle of *minimal inconsistency* proposed in [Pri89], and implemented by Upd_{mi}. The corresponding versions of $\boldsymbol{B}$ are denoted by $\boldsymbol{B}_{ex}$ and $\boldsymbol{B}_{mi}$.

For instance, if we have the above KB, and we then learn that Susan or Linda is blonde, we obtain:

$$\begin{aligned}Y_2 &= \mathsf{Upd}_{mi}(Y_1, b(S) \vee b(L))\\ &= \{\,\{b(L), l(P,L), \sim b(S)\},\ \{b(L), l(P,S), \sim b(S)\}\,\}\end{aligned}$$

Thus, the query whether Peter likes someone who is either blonde or (definitely) not blonde is now answered positively:

$$Y_2 \vdash \exists x : l(P,x) \wedge (b(x) \vee \sim b(x))$$

2.4 Some Formal Properties of Knowledge Systems

The following is a list of some fundamental properties a KS may have. The first two conditions of Contraction and Permutation are well-known as so-called *structural rules* in Gentzen-style sequent systems. In a KS, they describe the behaviour of the update operation.

Contraction

$$\mathsf{Upd}(\mathsf{Upd}(\mathrm{KB}, A), A) = \mathsf{Upd}(\mathrm{KB}, A)$$

Permutation

$$\mathsf{Upd}(\mathsf{Upd}(\mathrm{KB}, A), B) = \mathsf{Upd}(\mathsf{Upd}(\mathrm{KB}, B), A)$$

Both Contraction and Permutation follow from the property of

Update Synchronicity $\quad \mathsf{Upd}(\mathsf{Upd}(\mathrm{KB}, A), B) = \mathsf{Upd}(\mathrm{KB}, A \cup B)$

which expresses the fact that two inputs in succession (i.e. at different time points) can be handled as one aggregated input implying that the order of inputs does not matter.

Update Monotonicity

$$\mathrm{KB}_1 \leq \mathrm{KB}_2 \ \Rightarrow\ \mathsf{Upd}(\mathrm{KB}_1, A) \leq \mathsf{Upd}(\mathrm{KB}_2, A)$$

Lemma Redundancy (alias: Cut, Transitivity)

$$\mathrm{KB} \vdash F \ \ \&\ \ \mathsf{Upd}(\mathrm{KB}, F) \vdash G \ \Rightarrow\ \mathrm{KB} \vdash G$$

	$\boldsymbol{A}^+$	$\boldsymbol{A}$	$T\boldsymbol{A}$	$\boldsymbol{F}$	$\boldsymbol{F}_d$	$\boldsymbol{B}$	$\boldsymbol{B}_{ex}$	$\boldsymbol{B}_{mi}$
Contraction	√	√	√	√		√	√	√
Permutation	√		√			√	√	
Update Synchronicity	√		√			√	√	
Update Monotonicity	√	√	√	√		√	√	
Cumulativity	√	√	√	√	√	√	√	√
Monotonicity	√					√	√	√

Table 1: Formal properties of some basic knowledge systems.

Lemma Compatibility (alias Cautious Monotonicity, due to [Gab85])

$$\mathrm{KB} \vdash F \;\;\&\;\; \mathrm{KB} \vdash G \;\Rightarrow\; \mathsf{Upd}(\mathrm{KB}, F) \vdash G$$

Lemma Redundancy and Compatibility can be combined in the following condition of

Cumulativity $\quad \mathrm{KB} \vdash F \;\Rightarrow\; C(\mathsf{Upd}(\mathrm{KB}, F)) = C(\mathrm{KB})$

Even stronger than Cumulativity is the following property proposed in [Gär88],

Vacuity $\quad \mathrm{KB} \vdash F \;\Rightarrow\; \mathsf{Upd}(\mathrm{KB}, F) = \mathrm{KB}$

2.5 Nonmonotonicity

The following condition of Monotonicity captures the idea that a system is considered *monotonic* if all consequences of a KB are preserved after it is updated by some new piece of information.

Monotonicity $\quad C(\mathrm{KB}) \subseteq C(\mathsf{Upd}(\mathrm{KB}, F))$

Though fundamental in the theory of consequence operations due to Tarski, this is too strong a requirement for knowledge systems in general.

There are two 'parameters' on which Monotonicity depends: the update operation may be ampliative, and the inference relation may be persistent.

Observation 1 *A KS is monotonic if it is ampliative and persistent (even if its update operation is nonmonotonic).*

For instance, $\boldsymbol{A}$ is nonmonotonic since it allows for non-persistent queries. On the other hand, $\boldsymbol{B}_{mi}$, admitting only for ampliative inputs and persistent queries, is monotonic even though its update operation Upd_{mi}, by virtue of its inconsistency handling mechanism, is nonmonotonic. For instance,

$$\mathsf{Upd}_{mi}(0, (p \vee q) \wedge {\sim} p) = \{\{q, {\sim} p\}\} \vdash q$$

while

$$\mathsf{Upd}_{mi}(\{\{{\sim} q\}\}, (p \vee q) \wedge {\sim} p) = \{\{p, {\sim} p, {\sim} q\},\ \{{\sim} p, q, {\sim} q\}\} \not\vdash q$$

3 Deductive Knowledge Bases

Deductive knowledge bases were introduced in [Wag94a] under the name 'rule knowledge bases'. Every knowledge system $\boldsymbol{K} = \langle L_{\text{KB}}, \vdash, L_{\text{Query}}, \mathsf{Upd}, L_{\text{Input}} \rangle$ can be inductively extended to a *deductive* knowledge system $D\boldsymbol{K}$. In $D\boldsymbol{K}$, a knowledge base $X \in L_{\text{KB}}$ is supplemented by a set $R \subseteq L^x_{\text{Input}} \times L^x_{\text{Query}}$ of deduction rules $r = \langle F, G \rangle$ with conclusions $F \in L^x_{\text{Input}}$ and premises $G \in L^x_{\text{Query}}$, also written as '$F \leftarrow G$', such that $\text{Free}(F) \subseteq \text{Free}(G)$, and G is evaluable. Rules of this form are called *range-restricted*, the set of such rules will be denoted by $R(L_{\text{Input}} \leftarrow L_{\text{Query}})$. Rules are interpreted as mappings between KBs, $r : L_{\text{KB}} \to L_{\text{KB}}$, since their application is defined as

$$r(X) := \mathsf{Upd}(X, \{F\sigma : \sigma \text{ is a ground substitution for } G \text{ such that } X \vdash G\sigma\})$$

Notice that in the case of a non-applicable rule, we get $r(X) = \mathsf{Upd}(X, \emptyset) = \mathsf{Upd}(X, \bigwedge\emptyset) = \mathsf{Upd}(X, 1) = X$.

Let $[R]$ denote the instantiation of R. For $r = F \leftarrow G$, we will also write C_r instead of the conclusion F, and P_r instead of the premise G. If $r \in [R]$, then

$$r(X) = \begin{cases} \mathsf{Upd}(X, C_r) & \text{if } X \vdash P_r \\ X & \text{otherwise} \end{cases}$$

In the sequel, if not otherwise noted, we shall identify R with its instantiation $[R]$.

Observation 2 *A ground rule is idempotent, $r(r(X)) = r(X)$, iff Upd satisfies Contraction.*

Proof: If $X \not\vdash P_r$, then by definition, $r(r(X)) = r(X) = X$. Otherwise, $r(r(X)) = r(\mathsf{Upd}(X, C_r))$. Then if $\mathsf{Upd}(X, C_r) \not\vdash P_r$, we get

$$r(r(X)) = r(\mathsf{Upd}(X, C_r)) = \mathsf{Upd}(X, C_r) = r(X)$$

Otherwise, by Contraction,

$$r(r(X)) = \mathsf{Upd}(\mathsf{Upd}(X, C_r), C_r) = \mathsf{Upd}(X, C_r) = r(X) \qquad \square$$

Observation 3 *If the 'facts' X of a deductive knowledge base $\langle X, R \rangle$ consist of a set of compiled input formulas $A_X \subseteq L_{\text{Input}}$, $X = \mathsf{Upd}(0, A_X)$, then the deductive knowledge base can be rewritten as a set of rules:*

$$\langle X, R \rangle \quad \longrightarrow \quad \langle 0, R \cup \{F \leftarrow 1 : F \in A_X\} \rangle$$

where those 'improper' rules with a trivially true premise, $F \leftarrow 1$, represent the 'facts'.

Definition 5 (Deductive Closure) *$Z \in L_{\text{KB}}$ is a* deductive closure *of $\langle X, R \rangle$ if it extends X, $Z \geq X$, and if it is closed under all rules of R: $r(Z) = Z$ for all $r \in R$.*

Instead of 'deductive closure' we shall also just say 'closure'.

The semantics of a deductive knowledge base $\langle X, R\rangle$ is determined by the definition of the notion of a preferred closure. In general, there may be several preferred closures, or none. We denote their collection by $R(X)$, and write $R(X) \vdash F$ as an abbreviation of $\forall Z \in R(X) : Z \vdash F$. If $\langle X, R\rangle$ has several preferred closures, a valid consequence must be inferrable from all of them:

$$\langle X, R\rangle \vdash_d F \;:\Longleftrightarrow\; R(X) \vdash F$$

In the specific case where all rules $r \in R$ are monotonic and ampliative, there is exactly one preferred closure of $\langle X, R\rangle$, namely the informationally least one. In the general case, we shall assume that the *stable closures* (see below) are the preferred ones. All these cases will be treated separately in the following sections.

The input language of $D\boldsymbol{K}$ is the same as that of $\boldsymbol{K}$, and we have

$$\mathsf{Upd}_d(\langle X, R\rangle,\, F) := \langle \mathsf{Upd}(X, F),\, R\rangle$$

Formally,

$$D\boldsymbol{K} \;:=\; \langle\, L_{\text{KB}} \times 2^{R(L_{\text{Input}} \leftarrow L_{\text{Query}})},\, \vdash_d,\, L_{\text{Query}},\, \mathsf{Upd}_d,\, L_{\text{Input}}\,\rangle$$

In the sequel, we will omit the subscript d.

Example 3 (DDBs) *Deductive knowledge bases of $D\boldsymbol{A}$ correspond to deductive databases (DDBs), resp. normal logic programs. For instance, the program*

$$\Pi_1 = \{r(S),\, l(P, L),\, l(y, x) \leftarrow l(x, y)\},$$

corresponding to the deductive knowledge base $\langle X_1, R_1\rangle$ with

$$X_1 = \{r(S),\, m(P, L)\}, \text{ and } R_1 = \{m(y, x) \leftarrow m(x, y)\},$$

resp. to $\langle X_2, R_2\rangle$ with

$$X_2 = \{\}, \text{ and } R_2 = \{r(S) \leftarrow 1,\, l(P, L) \leftarrow 1,\, l(y, x) \leftarrow l(x, y)\},$$

has the unique minimal closure

$$R_1(X_1) = R_2(X_2) = \{r(S),\, l(P, L),\, l(L, P)\}$$

Example 4 (ELPs) *An extended logic program (ELP, see [GL90]) corresponds to a deductive factbase in the system $D\boldsymbol{F}$. In deductive factbases, one can express default rules by combining both kinds of negation. For instance, the rule*

$$f(x) \leftarrow b(x) \wedge -{\sim} f(x)$$

expresses the default that birds (normally) fly.

Example 5 (EDLPs) *An extended disjunctive logic program (EDLP, see [MR93]) without negation-as-failure corresponds to a deductive epistemic state. For instance,*

$$\Pi_2 = \{b(L),\, l(P, L) \vee l(P, S),\, {\sim} b(S),\, b(x) \leftarrow l(P, x)\}$$

corresponds to $\langle Y_2, R_2\rangle$, where Y_2 is as in section 2.3.4, and $R_2 = \{b(x) \leftarrow l(P, x)\}$, expressing that every woman Peter likes is blonde. In $D\boldsymbol{B}_{mi}$, $R_2(Y_2) = \{\{b(L), l(P, L), {\sim} b(S)\}\}$.

Definition 6 *A mapping $f : A \to A$ from a preorder $\langle A, \leq \rangle$ into itself is called* monotonic *if $f(x) \leq f(y)$ whenever $x \leq y$. It is called* ampliative *if $x \leq f(x)$. A rule is called* monotonic *(resp.* ampliative*) if it is a monotonic (resp. ampliative) mapping. A rule is called* persistent *if its premise is a persistent query formula. A rule knowledge base $\langle X, R \rangle$ is called persistent (ampliative, monotonic) if all rules $r \in R$ are persistent (ampliative, monotonic).*

Observation 4 *If Update Monotonicity holds, a rule is monotonic whenever it is a) both persistent and ampliative, or b) both anti-persistent and reductive.*

Proof of a): Let $X_1 \leq X_2$. We have to distinguish three cases. The first case, where r is neither applicable in X_1 nor in X_2, is trivial. If r is not applicable in X_1 but in X_2, then by Ampliative Update:

$$r(X_1) = X_1 \leq X_2 \leq \mathsf{Upd}(X_2, C_r) = r(X_2)$$

Otherwise, by Update Monotonicity and Persistent Inference:

$$r(X_1) = \mathsf{Upd}(X_1, C_r) \leq \mathsf{Upd}(X_2, C_r) = r(X_2) \qquad \square$$

Rules of positive logic programs, for instance, are persistent, ampliative and monotonic. Normal logic programs consist of ampliative rules, while active databases, resp. production rule systems such as OPS5 where the application of rules may cause the deletion of information, allow also for non-ampliative rules.

Definition 7 (Supported Closure) *A closure Z of a deductive knowledge base $\langle X, R \rangle$ is called* supported *if there is an ordinal α and a sequence of rules $(r_i)_{1 \leq i < \alpha} \subseteq [R]$, such that its composition, $r = \bigcirc_{\alpha > i \geq 1} r_i$, computes Z: $Z = r(X)$.*[15]

Example 6 *In $\boldsymbol{TA}$, let $X = \{ r(c)@3,\ q(c)@[2,5] \}$, and $R = \{ p(x)@t \leftarrow q(x)@t \wedge -r(x)@t \}$. Then the only supported closure is*

$$R(X) = \{ r(c)@3,\ q(c)@[2,5],\ p(c)@[(2,2),(4,5)] \}$$

Other minimal closures, such as $\{ r(c)@[3,5],\ q(c)@[2,5],\ p(c)@2 \}$, are not supported.

We have to explain how the composition $r = \bigcirc_{\alpha > i \geq 1} r_i$ is defined in the infinite case, i.e. when α is an infinite ordinal. In this case we shall make two restrictive assumptions: 1) that the information ordering $\langle L_{\mathrm{KB}}, \leq \rangle$ is a complete lattice (being the case, for instance, in $\boldsymbol{DA}$ and in $\boldsymbol{DF}$), and 2) that all rules in R are ampliative. We now inductively define $(X_i)_{1 \leq i < \alpha} \subseteq L_{\mathrm{KB}}$ as follows:

1. $X_1 := X$.

2. Assume that X_i has been defined for $i < \beta$, and $X_i \leq X_j$ whenever $i \leq j < \beta$.

[15] Notice that this definition of supportedness differs from that one introduced in [ABW88] by capturing a kind of grounded bottom-up support rather than the non-grounded top-down support captured there.

(a) If β is a successor ordinal, then $X_{\beta-1}$ exists and we can define

$$X_\beta := r_{\beta-1}(X_{\beta-1})$$

Since $r_{\beta-1}$ is ampliative, $X_{\beta-1} \leq X_\beta$.

(b) If β is a limit ordinal, then we define

$$X_\beta := \sup_{i<\beta} X_i$$

This supremum exists since it is assumed that the information ordering is a complete lattice.

Thus, in the infinite case, under the above assumptions, we get

$$\bigcirc_{\alpha>i\geq 1} r_i(X) = \sup_{i<\alpha} X_i$$

as a supported closure, while in the finite case we have

$$\bigcirc_{\alpha>i\geq 1} r_i(X) = r_{\alpha-1} \circ \ldots \circ r_2 \circ r_1(X)$$

If $Z = \bigcirc_{\alpha>i\geq 1} r_i(X)$ is a supported closure of $\langle X, R\rangle$, then we may assume (without loss of generality) that for every $i < \alpha$, we have $r_i \circ \ldots \circ r_2 \circ r_1(X) \vdash P_{r_{i+1}}$, and

$$Z = X + \sum_{1\leq i<\alpha} C_{r_i}$$

respectively, if Update Synchronicity holds,

$$Z = X + \{C_{r_i} : 1 \leq i < \alpha\}$$

In general, however, we may not conclude that $Z \vdash P_{r_i}$ for all $i \in [1, \alpha)$.

Not all supported closures are acceptable. There are even cases of minimal supported closures which do not correspond to our intuition of an acceptable, or, in other words, intended closure as the following example shows.

Example 7 *In $D\boldsymbol{A}$, let $X = \{s\}$, and $R = \{r \leftarrow s,\ q \leftarrow -r,\ p \leftarrow -q\}$. Both $\{q, r, s\}$, and $\{p, r, s\}$ are minimal supported closures, but only the latter one is an intended closure.*

We shall prefer closures which are *stable* in the sense that each rule application within the computation sequence preserves the applicability of all previously applied rules. The notion of a *stable closure* was introduced in [Wag94b].

Definition 8 (Stable Closure) *A closure Z of a deductive knowledge base $\langle X, R\rangle$ is called* stable *if there is a sequence of rules $\{r_i \in R : 1 \leq i < \alpha\}$, such that its composition, $r = \bigcirc_{\alpha>i\geq 1} r_i$, forms a stable computation of Z in the following sense: $Z = r(X)$, and*

If $r_{k-1} \circ \ldots \circ r_1(X) \vdash P_{r_k}$, then $r_n \circ \ldots \circ r_1(X) \vdash P_{r_k}$, for all n, k with $\alpha > n \geq k \geq 1$.

If $Z = \bigcirc_{\alpha > i \geq 1} r_i(X)$ is a stable closure of $\langle X, R\rangle$, then we may assume (without loss of generality) that for every $i < \alpha$, and $j \leq i+1$,

$$r_i \circ \ldots \circ r_2 \circ r_1(X) \vdash P_{r_j}$$

implying that $Z \vdash P_{r_j}$ for all $j \in [1, \alpha)$.

Definition 9 (Preferred Closures) *We define the preferred, or intended, closures of a deductive knowledge base to be the minimally inconsistent stable ones:*

$$R(X) := \mathrm{MInc}(\{Z \in L_{\mathrm{KB}} : Z \textit{ is a stable closure of } \langle X, R\rangle\})$$

In the sequel, we shall identify the singleton $R(X) = \{Z\}$ with its single element Z, i.e. we shall write $R(X)$ instead of Z.

Example 8 *The deductive factbase* $\langle \emptyset,\ \{\sim q \leftarrow -\sim p,\ \sim p \leftarrow -q\}\rangle$ *has exactly one stable closure:* $\{\sim p\}$.

Example 9 *In* $\boldsymbol{DF}$, *let* $X = \{\sim p\}$, *and* $R = \{\, p \leftarrow -q,\ q \leftarrow -p \,\}$. *Then* $R(X) = \{\{\sim p, q\}\}$.

If a deductive knowledge base does not have any stable closure, it is called *unstable.*

Example 10 *In* $\boldsymbol{DF}_d$, *the deductive factbase* $\langle \{\sim p\},\ \{\, q \leftarrow p \vee -p,\ p \leftarrow q \,\}\rangle$ *is unstable, while in* $\boldsymbol{DF}$ *it has a stable closure:* $\{\sim p, q, p\}$.

4 The Immediate Consequence Operator

Definition 10 *For any deductive knowledge base* $\langle X, R\rangle$, *and any* $Z \in L_{\mathrm{KB}}$, *we define*

$$T_{X,R}(Z) := \mathsf{Upd}(X, \{F : F \leftarrow G \in [R] \ \ \& \ \ Z \vdash G\})$$

and its cumulative version

$$T_R(Z) := T_{Z,R}(Z)$$

for which the following iteration sequence can be formed:

$$X^0 = X, \qquad X^{i+1} = T_R(X^i) \quad (i \geq 0)$$

In the case of positive logic programs, i.e. for $\boldsymbol{DA}^+$, one obtains the classical T_P-operator: let $\langle X, R\rangle$ be the DKB corresponding to a program P, i.e. $X = \{a : a \leftarrow \emptyset \in P\}$, and $R = \{a \leftarrow A \in P : A \neq \emptyset\}$, then for $I \subseteq \mathrm{At}$,

$$T_P(I) = \{a : a \leftarrow A \in [P] \ \ \& \ \ I \models A\}$$

as defined by Van Emden and Kowalski in [vEK76], is equal to $T_{X,R}(I)$.

Observation 5 *If Update Synchronicity holds, the fixpoints of* T_R *are exactly the KBs closed under* R.

Proof: Let Z be a closure, then for any rule $r \in R$ we have $r(Z) = Z$. Let us show that

$$\mathsf{Upd}(Z, \{C_r : Z \vdash P_r\}) = Z$$

For every r such that $Z \vdash P_r$, we have $\mathsf{Upd}(Z, C_r) = Z$. Therefore, by Update Synchronicity, we get the desired conclusion.

Conversely, if Z is a fixpoint of T_R, then $\mathsf{Upd}(Z, \{C_r : Z \vdash P_r\}) = Z$. This clearly implies, by Update Synchronicity, that $\mathsf{Upd}(Z, C_r) = Z$, whenever $Z \vdash P_r$. Therefore, Z is closed under R. □

Observation 6 *If Contraction holds, the fixpoints of $T_{X,R}$ are also fixpoints of T_R.*

Proof: Let Z be a fixpoint of $T_{X,R}$, then we have

$$Z = T_{X,R}(Z) = \mathsf{Upd}(X, \{C_r : r \in R \ \ \& \ \ Z \vdash P_r\})$$

By Contraction we get

$$\mathsf{Upd}(Z, A) = \mathsf{Upd}(\mathsf{Upd}(X, A), A) = \mathsf{Upd}(X, A) = Z$$

where $A = \{C_r : r \in R \ \ \& \ \ Z \vdash P_r\}$. □

Observation 7 *If Z is closed under R such that for some $A \subseteq \{C_r : r \in R \ \ \& \ \ Z \vdash P_r\}$, $Z = \mathsf{Upd}(X, A)$, then Z is a fixpoint of $T_{X,R}$.*

Proof: Let Z be closed under R such that $Z = \mathsf{Upd}(X, A)$ where $A \subseteq \{C_r : r \in R \ \ \& \ \ Z \vdash P_r\}$. Let $B = \{C_r : r \in R \ \ \& \ \ Z \vdash P_r\} - A$. By Update Synchronicity, we get

$$Z = \mathsf{Upd}(Z, B) = \mathsf{Upd}(\mathsf{Upd}(X, A), B) = \mathsf{Upd}(X, \{C_r : r \in R \ \ \& \ \ Z \vdash P_r\})$$

and consequently, $Z = T_{X,R}(Z)$. □

Claim 1 *If Update Synchronicity holds, then every stable closure of $\langle X, R\rangle$ is a fixpoint of $T_{X,R}$ and T_R.*

Proof: Let Z be a stable closure of $\langle X, R\rangle$. Then for some set of rules $\{r_i : 1 \leq i < \alpha\} \subseteq R$,

$$Z = \mathsf{Upd}(X, \{C_{r_i} : 1 \leq i < \alpha\})$$

By observation 7, Z is a fixpoint of $T_{X,R}$. By observation 6, Z is also a fixpoint of T_R. □

5 Monotonic DKBs

Recall that a deductive knowledge base $\langle X, R\rangle$ is monotonic if all $r \in R$ are monotonic mappings. This is, e.g., the case if Update Monotonicity holds, and all $r \in R$ are persistent and ampliative (see observation 4).[16]

For instance, all deductive temporal databases whose rules do not contain the weak negation $-$, such as $p(x)@T \leftarrow (q(x) \vee \exists y r(x,y))@T \wedge p(x)@2$, are monotonic. In $\boldsymbol{B}_{mi}$, although Update Monotonicity does not hold, rules with definite conclusions are monotonic. On the other hand, there are no monotonic rules at all in $\boldsymbol{DF}_d$, and hence there are no monotonic deductive defeasible factbases.

More exoticaly, anti-persistent reductive rules, such as the $\boldsymbol{DA}$ rule $-p \leftarrow -q$, are also monotonic if Update Monotonicity holds.

Claim 2 *A monotonic DKB has at most one supported closure which is its least closure.*

Proof: Assume that $\hat{X} = \bigcirc_{\alpha > i \geq 1} r_i(X)$ is a supported closure of $\langle X, R\rangle$. Let Z be any deductive closure of $\langle X, R\rangle$. Since r_1 is monotone, we get

$$r_1(X) \leq r_1(Z) = Z$$

It is obvious that we can get

$$\bigcirc_{\beta \geq i \geq 1} r_i(X) \leq Z$$

for every $\beta < \alpha$ in this way. Consequently, $\hat{X} \leq Z$. $\Box$

6 Persistent Ampliative DKBs

Recall that persistent ampliative DKBs are monotonic if Update Monotonicity holds. If a knowledge system violates Update Monotonicity, such as $\boldsymbol{F}_d$ and $\boldsymbol{B}_{mi}$, rules may be nonmonotonic even if they are persistent and ampliative. Notice that there are no ampliative rules at all in $\boldsymbol{DF}_d$.

Let R be persistent, and Z_1 and Z_2 be two fixpoints of $T_{X,R}$, such that $Z_1 \leq Z_2$. Because of Persistence, $Z_1 \vdash P_r$ implies that $Z_2 \vdash P_r$. Therefore,

$$\{C_r : r \in R \ \ \& \ \ Z_1 \vdash P_r\} \subseteq \{C_r : r \in R \ \ \& \ \ Z_2 \vdash P_r\}$$

and consequently, provided that Update Synchronicity holds,

$$\begin{aligned} Z_2 &= \mathsf{Upd}(X, \{C_r : r \in R \ \ \& \ \ Z_2 \vdash P_r\}) \\ &= \mathsf{Upd}(X, \{C_r : Z_1 \vdash P_r\} \cup \{C_r : Z_2 \vdash P_r \ \ \& \ \ Z_1 \not\vdash P_r\}) \\ &= \mathsf{Upd}(\mathsf{Upd}(X, \{C_r : Z_1 \vdash P_r\}), \{C_r : Z_2 \vdash P_r \ \ \& \ \ Z_1 \not\vdash P_r\}) \\ &= \mathsf{Upd}(Z_1, \{C_r : Z_2 \vdash P_r \ \ \& \ \ Z_1 \not\vdash P_r\}) \end{aligned}$$

[16]Notice that we do not require in this section that the KS is monotonic, but only that the DKB is monotonic.

Claim 3 *If Update Synchronicity holds, and R is persistent and ampliative, the operator $T_{X,R}$ is monotone.*

Proof: Let $Z_1 \leq Z_2$. Because all premises of R are persistent we have

$$\{C_r : r \in R \ \& \ Z_1 \vdash P_r\} \subseteq \{C_r : r \in R \ \& \ Z_2 \vdash P_r\}$$

which implies

$$\begin{aligned} T_{X,R}(Z_2) &= \mathsf{Upd}(X, \{C_r : r \in R \ \& \ Z_2 \vdash P_r\}) \\ &= \mathsf{Upd}(T_{X,R}(Z_1), \{C_r : r \in R \ \& \ Z_2 \vdash P_r \ \& \ Z_1 \not\vdash P_r\}) \end{aligned}$$

Since all conclusions of R are ampliative, we get $T_{X,R}(Z_1) \leq T_{X,R}(Z_2)$. □

Claim 4 *Any supported closure of a persistent ampliative DKB is stable.*

Proof: Let $Z = \bigcirc_{\alpha > i \geq 1} r_i(X)$ be a supported closure of $\langle X, R\rangle$. Assume that the rules $(r_i)_{1 \leq i < \alpha}$ are ampliative and persistent. Abbreviating $X_j := \bigcirc_{j > i \geq 1} r_i(X)$, we can assume, without loss of generality, that for every $j < \alpha$, we have $X_j \vdash P_{r_j}$. Clearly, this implies that

$$X_{j+1} = \mathsf{Upd}(X_j, C_{r_j}) = X + \sum_{1 \leq i \leq j} C_{r_j}$$

Since all inputs C_{r_j} are ampliative, we get $X_j \geq X_i$ for all $i \leq j$. Because all premises P_{r_j} are persistent, $X_j \vdash P_{r_i}$, for all $i \leq j$. □

Assume that r_1 and r_2 are ampliative and persistent, and $Z \vdash P_{r_i}$ for $i = 1, 2$. If Update Synchronicity holds, we get

$$r_1 \circ r_2(Z) = r_2 \circ r_1(Z) = \mathsf{Upd}(Z, \{C_{r_1}, C_{r_2}\})$$

Therefore, we conclude that if the rules $(r_i)_{1 \leq i < \alpha}$ are ampliative and persistent, and $Z \vdash P_{r_i}$ for $1 \leq i < \alpha$, then to compute $\bigcirc_{\alpha > i \geq 1} r_i(Z)$, the order in which the rules are applied is not important. Therefore, under the assumption of Update Synchronicity, if $\langle X, R\rangle$ is ampliative and persistent and

$$R' \subseteq R_Z := \{r \in R : Z \vdash P_r\}$$

we can write $\bigcirc R'(Z)$ for the composition of all the rules in R' applied to Z without specifying the order in which these rules are applied. Moreover we have $\bigcirc R'(Z) = \mathsf{Upd}(Z, \{C_r : r \in R'\})$.

Since for a persistent and ampliative DKB $\langle X, R\rangle$ the operator $T_{X,R}$ is monotone, the corresponding fixpoint set $Fix(T_{X,R})$ is a complete sublattice. Hence there exists a least fixed point of $T_{X,R}$ denoted $\hat{X}$. The general theory implies that

$$\hat{X} = \lim T^i_{X,R}(X)$$

It is easy to show then, under the assumption of Update Synchronicity, that the iterates of $T_{X,R}$ are equal to the iterates of T_R, i.e. for every i we have

$$T^i_{X,R}(X) = T^i_R(X) = X^i$$

First notice that by definition, $T_{X,R}(X) = T_R(X) = X^1$. We want to prove that $X^{i+1} = \mathsf{Upd}(X, \{C_r : X^i \vdash P_r\})$. Let us show it for $i = 2$. We have $X \leq X^1$ (because of ampliativity), then if $X \vdash P_r$, we have $X^1 \vdash P_r$. Hence

$$\begin{aligned}
X^2 &= T_R(X^1) = \mathsf{Upd}(X^1, \{C_r : X^1 \vdash P_r\}) \\
&= \mathsf{Upd}(\mathsf{Upd}(X, \{C_r : X \vdash P_r\}), \{C_r : X^1 \vdash P_r\}) \\
&= \mathsf{Upd}(X, \{C_r : X \vdash P_r\} \cup \{C_r : X^1 \vdash P_r\}) \\
&= \mathsf{Upd}(X, \{C_r : X^1 \vdash P_r\}) \\
&= T_{X,R}(X^1)
\end{aligned}$$

since $\{C_r : X \vdash P_r\} \subset \{C_r : X^1 \vdash P_r\}$. In the general case the proof is identical using the fact that the sequence X^i is monotone. Consequently, we get the following:

Claim 5 *Let $\langle X, R\rangle$ be a persistent ampliative DKB. Then, if Update Synchronicity holds,*

$$\hat{X} = \lim T^i_{X,R}(X) = \lim X^i = \sup X^i$$

We may ask if there is a relationship between $\hat{X}$ and the supported closures. The answer is given by the following result.

Claim 6 *In a KS satisfying Update Synchronicity and where the information ordering is a partial order, any supported closure of a persistent ampliative DKB is equal to $\hat{X}$. Therefore every persistent ampliative DKB has a unique supported closure.*

Proof: Let Z be a supported closure of a persistent ampliative DKB $\langle X, R\rangle$. We know that Z is stable. Write

$$Z = \bigcirc_{\alpha > i \geq 1} r_i(X) = \mathsf{Upd}(X, \{C_{r_i} : i < \alpha\})$$

with $Z \vdash P_{r_i}$ for every $i < \alpha$. Since Z is stable, then Z is a fixpoint of $T_{X,R}$. Therefore we have $\hat{X} \leq Z$. Let us prove that $Z \leq \hat{X}$. Since $X \vdash P_{r_1}$ and $X^1 = \mathsf{Upd}(X, \{C_r : X \vdash P_r\})$, then $r_1(X) \leq X^1$. Using the properties of the sequence X^i, one can easily prove that

$$\bigcirc_{\beta \geq i \geq 1} r_i(X) \leq X^\beta$$

for every $\beta < \alpha$. Therefore

$$Z = \sup_{\beta < \alpha} \bigcirc_{\beta \geq i \geq 1} r_i(X) \leq \hat{X} \qquad \square$$

Notice that $\hat{X}$ is not the least closure, in general. However if the rules are monotone, it is the least closure.

7 Monotonic Ampliative DKBs

Observation 8 *The composition of two monotonic ampliative mappings r_1 and r_2 will always provide an upper bound for both of them: $r_1 \circ r_2 \geq r_1, r_2$, resp.*

$$r_1(r_2(X)) \geq r_i(X) \quad \textit{for all } X \in L_{\mathrm{KB}}, \textit{ and } i = 1, 2$$

Proof: Because r_1 is ampliative, $r_1(r_2(X)) \geq r_2(X)$. It remains to show that $r_1(r_2(X)) \geq r_1(X)$. If r_2 is not applicable, then $r_1(r_2(X)) = r_1(X) \geq X = r_2(X)$. Otherwise, since r_2 is ampliative, $r_2(X) \geq X$, and by monotonicity,

$$r_2(X) \geq X \;\Rightarrow\; r_1(r_2(X) \geq r_1(X) \qquad \Box$$

Therefore, the set of all compositions of rules from a monotonic ampliative rule set R is a directed set. This can be used, under the further assumption of continuity, to apply Scott's theory of continuos lattices [Sco72] in order to obtain the intended closure of $\langle X, R\rangle$, as proposed in [Bel77]. We are, however, more interested here to study the existence of intended closures under assumptions weaker than continuity.

Claim 7 *A monotonic ampliative DKB has a unique supported closure which is the least closure.*

Proof: Let $\langle X, R\rangle$ be a monotonic ampliative DKB. Since the rules are ampliative, there exists a supported closure $\hat{X}$. Let Z be a closure such that $X \leq Z$. Then for any rule r we have $r(X) \leq r(Z) = Z$. Therefore, we have $\hat{X} \leq Z$. $\Box$

We do not know whether $\hat{X}$ is stable. It is clear that $\hat{X}$ is the least fixpoint of T_R since the fixpoints of T_R are exactly the closures. But we do not know whether $\hat{X}$ is the least fixpoint of $T_{X,R}$. Notice that in this case it is unclear whether $T_{X,R}$ is monotone while it is obvious for T_R. Therefore, we have two theories: one for persistent ampliative DKBs and the other one for monotonic ampliative DKBs. They can be combined in the case of persistent ampliative DKBs of a KS satisfying Update Monotonicity.

8 Ampliative DKBs

Typical examples for non-persistent ampliative DKBs are all kinds of logic programs (normal, extended, disjunctive, epistemic, etc.) where non-persistent operators, such as negation-as-failure or epistemic modalities, are allowed in the premise of a rule but not in the conclusion.

Claim 8 *Every ampliative deductive knowledge base has a supported closure if Update Synchronicity holds (in the infinite case we also need to assume that the information ordering is a complete lattice).*

Proof: Let $\langle X, R\rangle$ be an ampliative DKB. Set $X_1 := X$. If for all $r \in R$ we have $r(X_1) = X_1$, we set $r_1 := Id$ (the identity map). Otherwise we choose $r_1 \in R$ such that $r_1(X_1) \neq X_1$. Assume that $(r_i)_{1 \leq i < \alpha}$ has been constructed. We set

$$X_\alpha := \bigcirc_{\alpha > i \geq 1} r_i(X)$$

where α is an ordinal. If for all $r \in R$ we have $r(X_\alpha) = X_\alpha$, we set $r_\alpha := Id$. Otherwise we choose $r_\alpha \in R$ such that $r_\alpha(X_\alpha) \neq X_\alpha$. Since all $r \in R$ are ampliative, (X_α) is monotonically increasing, and according to ZF set theory, there is an ordinal $\beta < \gamma$, such that $X_\beta = X_{\beta+1}$, and γ corresponds to the well-ordering type of R.

It remains to show that X_β is closed under r_i for $i \leq \beta$, since it is closed trivially under all other rules $r \in R - \{r_i : 1 \leq i \leq \beta\}$, being not applicable. Clearly, for every $i < \beta$, we have $X_{i+1} = r_i(X_i) = \mathsf{Upd}(X_i, C_{r_i})$. Therefore, if $X_\beta \not\vdash P_{r_i}$, then X_β is trivially closed under r_i. Otherwise,

$$\begin{aligned} r_i(X_\beta) &= r_i(\mathsf{Upd}(X, \{C_{r_j} : 1 \leq j < \beta\})) \\ &= \mathsf{Upd}(\mathsf{Upd}(X, \{C_{r_j} : 1 \leq j < \beta\}), C_{r_i}) \\ &= \mathsf{Upd}(X, \{C_{r_j} : 1 \leq j < \beta\}) \\ &= X_\beta \qquad \square \end{aligned}$$

Thus, the situation of ampliative DKBs is similar to that of normal logic programs. Although they always have supported closures, resp. minimal models, we are rather interested in their stable closures, resp. models. Therefore, an important question is: under which conditions does an ampliative DKB have a unique stable closure ? Can we generalize the logic programming notions of *acyclicity* [AB90] and *stratification* [ABW88, Prz88, PP90, BF91] for deductive knowledge bases ? This will be the subject of a sequel paper.

9 Non-Ampliative DKBs

Little is known on non-ampliative DKBs. And it does not seem to be clear whether non-ampliative deduction rules can be useful at all in knowledge representation (whereas it is obvious that action rules may be non-ampliative). We only discuss some examples.

Example 11 (Reductive Rules) *Rules in $\boldsymbol{DA}$ can have weakly negated conclusions, representing reductive input. For instance, we could have $r_1 = -q \leftarrow p$, $r_2 = s \leftarrow p$, and $R = \{r_1, r_2\}$. If applied to $X = \{p\}$, we obtain $R(X) = \{p, s\}$. The non-ampliative rule r_1 is redundant in this case: $R(X) = r_2(X)$. On the other hand, certain non-ampliative DKBs with reductive rules, like*

$$\langle X, R \rangle = \langle \{p, q\},\ \{-q \leftarrow p\} \rangle$$

do not have any closure, and are in this sense unsatisfiable. Reductive rules, therefore, do not seem to make sense in deductive knowledge bases: either they are redundant, or unsatisfiable.

Example 12 (Non-Ampliative Non-Reductive Rules) *In $\boldsymbol{F}_d$, inputs are neither ampliative nor reductive. Thus, we could have the following non-ampliative non-reductive rule set $R = \{q \leftarrow -p\}$, which is non-redundant in cases like $X = \emptyset$,*

$$R(\{\emptyset\}) = \{\{q\}\}$$

and unsatisfiable in certain other cases like $X = \{\sim q\}$, which cannot be closed under R.

10 Conclusion

The framework of deductive knowledge systems allows the formulation of a general theory of (possibly nonmonotonic) rules. We have classified certain important cases of deductive knowledge bases and presented some results on the existence of closures for them. Although several problems had to be left open, we have provided the theoretical foundations for many new extensions of logic programming and deductive database systems.

References

[ABW88] K.R. Apt, H. Blair and A. Walker: Towards a Theory of Declarative Knowledge, in J. Minker (Ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, 1988.

[AB90] K.R. Apt and M. Bezem: Acyclic Programs, *Proc. ICLP 1990*, MIT Press, 1990.

[BG93] C. Baral and M. Gelfond: Logic Programming and Knowledge Representation, Technical Report, University of Texas at El Paso, 1993.

[Bel77] N.D. Belnap: A Useful Four-valued Logic, in G. Epstein and J.M. Dunn (Eds.), *Modern Uses of Many-valued Logic*, Reidel 1977, 8–37.

[BF91] N. Bidoit and C. Froidevaux: Negation By Default and Unstratifiable Logic Programs, *Theoretical Computer Science* **78** (1991), 85–112.

[vEK76] M.H. van Emden and R.A. Kowalski: The Semantics of Predicate Logic as a Programming Language, *J. of the ACM* **23**:4 (1976), 733–742.

[Gab85] D. Gabbay: Theoretical Foundations for Nonmonotonic Reasoning in Expert Systems, in K.R. Apt (Ed.), *Proc. NATO Advanced Study Institute on Logics and Models of Concurrent Systems*, Springer Verlag, 1985, 439–457.

[Gär88] P. Gärdenfors: *Knowledge in Flux*, MIT Press, Cambridge, 1988.

[vGT91] A. van Gelder and R.W. Topor: Safety and Translation of Relational Calculus Queries, *ACM Transactions on Database Systems* 16:2 (1991), 235–278.

[GL88] M. Gelfond and V. Lifschitz: The Stable Model Semantics for Logic Programming, *Proc. ICLP 1988*, MIT Press, 1988.

[GL90] M. Gelfond and V. Lifschitz: Logic Programs with Classical Negation, *Proc. ICLP 1990*, MIT Press, 1990.

[GL91] M. Gelfond and V. Lifschitz: Classical Negation in Logic Programs and Disjunctive Databases, *J. New Generation Computing* **9** (1991), 365–385.

[Lev84] H.J. Levesque: Foundations of a Functional Approach to Knowledge Representation, *AI* **23**:2 (1984), 155-212.

[MR93] J. Minker and C. Ruiz: Semantics for Disjunctive Logic Programs with Explicit and Default Negation, *Proc. ISMIS'93*, Springer LNAI, 1993.

[Pea93] D. Pearce: 'Answer Sets and Constructive Logic, II: Extended Logic Programs and Related Nonmonotonic Formalisms, in L.M. Pereira and A. Nerode (Eds.), *Logic Programming and Nonmonotonic Reasoning*, MIT Press, 1993.

[Pri89] G. Priest: Reasoning about Truth, *AI* **39** (1989), 231–244.

[PP90] H. Przymusinska and T.C. Przymusinski: Weakly Stratified Logic Programs, *Fundamenta Informaticae* **13** (1990), 51–65.

[Prz88] T.C. Przymusinski: On the Declarative Semantics of Logic Programs with Negation, in J. Minker (Ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, 1988.

[Sco72] D. Scott: 'Continous Lattices: Toposes, Algebraic Geometry and Logic', Springer Lecture Notes in Mathematics **274** (1972), 97–136.

[Wag91] G. Wagner: A Database Needs Two Kinds of Negation, in H.-D. Gerhard and B. Thalheim (Eds.), *Proc. 3rd Int. Symp. on Mathematical Fundamentals of Database and Knowledge Base Systems MFDBS-91*, Springer LNCS **495** (1991), 357–371.

[Wag93] G. Wagner: Disjunctive Fact Bases and Logic Programming – Preliminary Report, in C. Baral and M. Gelfond (Eds.), *Proc. ILPS'93 Workshop on Logic Programming with Incomplete Information*, 1993.

[Wag94a] G. Wagner: *Vivid Logic – Knowledge-Based Reasoning with Two Kinds of Negation*, Springer LNAI **764** (1994).

[Wag94b] G. Wagner: Transforming Deductive into Active Databases, in N. Fuchs and G. Gottlob (Eds.), *Proc. of 10th Workshop on Logic Programming Zürich 1994*.

[Wag95a] G. Wagner: From Information Systems to Knowledge Systems, in E. Falckenberg (Ed.), *Proc. of IFIP Working Conf. on Information System Concepts (ISCO3)*, Chapman & Hall 1995.

[Wag95b] G. Wagner: Belnap's Epistemic States and Negation-as-Failure, in H. Wansing (Ed.), *Negation in Focus*, de Gruyter 1996.