Disjunctive, Deductive and Active Knowledge Bases

Gerd Wagner
gw@inf.fu-berlin.de

Preprint Version 1.0 October 1994

Contents

Chapter 1

General Introduction

In this report, we present a collection of papers in preprint status on recent issues in the area of knowledge representation and logic programming, together with some preliminary material on related topics. Chapter 3 on the operational semantics of rules is based on joint work together with M.A. Khamsi and D. Misane. Large parts of the work presented here have been done during an extended stay at the Institut de Recherche en Informatique de Toulouse (IRIT) on invitation of Luis Farinas del Cerro and Andreas Herzig to whom I want to thank for their hospitality.

The common framework of disjunctive, deductive and active knowledge bases consists of the theory of knowledge representation and reasoning systems proposed in [Wag94a]. The following novel ideas are presented in our report:

- Show that nonmonotonicity in knowledge systems has two sources: non-persistent queries and non-ampliative inputs, and in order to formalize this, one needs an information ordering for knowledge bases.
- Explain the meaning of negation-as-failure as an operator of the input language: it corresponds to deletion, resp. contraction.
- Investigate the semantics of negation-as-failure and the CWA on the basis of disjunctive information *without* rules.
- Explain the relationship between extended (disjunctive) logic programs, resp. systems with strong negation, and classical logic by means of the well-motivated concept of *exact predicates*.
- Clarify the difference between *definite* and *epistemic* failure by introducing appropriate modal operators.
- Show how deduction rules can be interpreted as update functions operating on knowledge bases. Establish an operational semantics of rules in this way which also applies to action rules.
- Show that, mathematically, the general problem of a semantics for a set of (possibly nonmonotonic) rules does not depend on any specific logical operator, such as negation-as-failure, but is rather a problem of defining appropriate common fixpoints for a set of (possibly nonmonotonic) functions. Therefore, the general version of the definition of the stable semantics should not refer to negation-as-failure (i.e. not use the Gelfond-Lifschitz transformation).

- Formalize the concept of action rules, and show its close relation to that of deduction rules.

Chapter 2

Knowledge Representation and Reasoning

The language of knowledge systems consists of the logical operator symbols $\wedge, \vee, \sim, -$ and 1 standing for conjunction, disjunction, strong negation, weak negation and the verum, respectively; the predicate symbols $p, q, r, \ldots$; the constant symbols $c, d, \ldots$ and variables $x, y, \ldots$. For the sake of simplicity we shall not consider functional terms but only variables and constants.[1] Further connectives, such as epistemic modalities, and the (possibly restricted) quantifiers $\exists$ and $\forall$ can be optionally added to the language.

An *atom* a is an atomic formula, it is called *proper*, if $a \neq 1$. *Literals* are either atoms or strongly negated atoms, $l = a | {\sim} a$. *Extended literals* are either literals or weakly negated literals, $e = l | {-l}$. We use $a, b, \ldots,$ $l, k, \ldots,$ $e, f, \ldots,$ and $F, G, H, \ldots$ as metavariables for atoms, literals, extended literals and formulas, respectively.[2] A variable-free expression is called *ground*. The set of all proper ground atoms (resp. literals, resp. extended literals) of a given language is denoted by At (resp. Lit, resp. XLit). If not otherwise stated, a formula is assumed to be ground. If *op-list* is a set of logical operators, say *op-list* $\subseteq \{1, -, \sim, \wedge, \vee, \rightarrow, \exists, \forall, \ldots\}$, then $L(\textit{op-list})$ denotes the respective set of well-formed formulas.

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 $E^- := \{l : -l \in E\}$.

Formulas of partial logic (with two kinds of negation) can be normalized in the same manner as those of classical logic. For this purpose, $\mathrm{DNS}(F)$, the *disjunctive normal set* of a formula F, is defined as follows:

$$
\begin{array}{lcl}
\mathrm{DNS}(1) & = & \{\emptyset\} \\
\mathrm{DNS}(e) & = & \{\{e\}\} \\
\mathrm{DNS}(F \wedge G) & = & \{E \cup D \,:\, E \in \mathrm{DNS}(F),\ D \in \mathrm{DNS}(G)\} \\
\mathrm{DNS}(F \vee G) & = & \mathrm{DNS}(F) \cup \mathrm{DNS}(G)
\end{array}
$$

All other cases of compound formulas can be handled by the following DeMorgan-style

[1] Therefore, we only deal with a finite Herbrand universe.

[2] I will frequently just say "literal" when I, precisely speaking, mean a proper ground literal.

rewrite rules:

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

The disjunctive normal form of a formula $G \in L(-, \sim, \wedge, \vee)$ is obtained as

$$\mathrm{DNF}(G) = \bigvee_{E \in \mathrm{DNS}(G)} \bigwedge E$$

2.1 Knowledge Systems

The inference relation between knowledge bases and query formulas is not uniform, in general. Only certain formulas may make sense for representing knowledge, that is, there will be a specific representation language L_{Repr}, and a knowledge base (KB) will be a 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 KS consists of two operations: an inference operation processing queries posed to the KB, and an update operation processing input formulas 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 add its information content to the KB.

In general, a 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, I will assume that a $\mathrm{KB} \in L_{\mathrm{KB}}$ is a finite set of expressions from a representation language, i.e. $L_{\mathrm{KB}} \subseteq 2^{L_{\mathrm{Repr}}}$.

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

$$\mathrm{KB}_1 \leq \mathrm{KB}_2 \quad \text{if } \mathrm{KB}_2 \text{ contains at least as much information as } \mathrm{KB}_1.$$

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).[3] The informationally empty KB will be denoted by 0. By definition, $0 \leq \mathrm{KB}$ for all KBs.

Hence, a KS is defined as a 7-tuple:[4]

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

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 certain properties describing their interaction (see below), and it holds that

[3] The usual way to compare the information content of two KBs in standard logic (see, e.g., the literature on belief revision), 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.

[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 described as a 'functional approach to knowledge representation'. In [Wag94a] the concept of KSs was further extended and proposed as an integrating framework offering a general perspective for knowledge representation and logic programming.

(KS1) $\mathrm{KB} \vdash 1$, and $\mathsf{Upd}(\mathrm{KB}, 1) = \mathrm{KB}$, for any $\mathrm{KB} \in L_{\mathrm{KB}}$.

(KS2) $\mathsf{Upd}(\mathrm{KB}, F) \vdash F$, for any $\mathrm{KB} \in L_{\mathrm{KB}}$, and for any $F \in L_{\mathrm{Input}} \cap L_{\mathrm{Query}}$ which is consistent with KB.

(KS3) $\langle L_{\mathrm{KB}}, \leq, 0\rangle$ is a preorder (called *information ordering*) with least element 0 (called *empty KB*), satisfying the following adequacy condition:

$$\mathrm{KB}_1 \leq\geq \mathrm{KB}_2 \quad \text{iff} \quad \begin{array}{l} \forall F \in L_{\mathrm{Input}} \forall G \in L_{\mathrm{Query}} : \\ \mathsf{Upd}(\mathrm{KB}_1, F) \vdash G \Leftrightarrow \mathsf{Upd}(\mathrm{KB}_2, F) \vdash G \end{array}$$

which requires that KBs are informationally indistinguishable iff they are *update equivalent.*[5]

A consequence operation C is defined as usual: $C(\mathrm{KB}) = \{F \in L_{\mathrm{Query}} : \mathrm{KB} \vdash F\}$. Also an answer operation, taking a knowledge base and an open query, and providing the corresponding set of valid answers, can be defined in terms of the inference relation. If the knowledge base is definite, an answer is a tuple, and the set of answers to a query is a relation:

$$\mathsf{Ans}(\mathrm{KB}, F(x)) := \{t : \mathrm{KB} \vdash F(t)\}$$

where x is a variable (tuple) and t a term, resp. constant (tuple). In general, however, answers may be indefinite, i.e. sets of possible answer substitutions corresponding to minimal disjunctive consequences:

$$\mathsf{Ans}(\mathrm{KB}, F(x)) := \min\{T : \mathrm{KB} \vdash \bigvee\{F(t) : t \in T\}\}$$

Since it appears reasonable to require that any information entered to a KB can be queried afterwards, we shall assume that $L_{\mathrm{Input}} \subseteq L_{\mathrm{Query}}$.

We shall also allow for updates by (possibly infinite) sets of input formulas, i.e. use Upd also to denote the more general update operation

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

such that for any finite $A \subseteq L_{\mathrm{Input}}$, $\mathsf{Upd}(\mathrm{KB}, A) = \mathsf{Upd}(\mathrm{KB}, \bigwedge A)$. We shall also write $\mathrm{KB} + F$ instead of $\mathsf{Upd}(\mathrm{KB}, F)$, resp.

$$\mathrm{KB} + \sum_{i=1}^{\alpha} F_i$$

instead of $(\ldots((\mathrm{KB} + F_1) + F_2) + \ldots) + F_\alpha$.

Example 1 (Relational Databases) *A KB consisting of ground atoms corresponds to a relational database (RDB). For instance, $X_1 = \{r(S), r(P), m(P, L)\}$ may represent the information that both Peter and Susan are residents, and that Peter is married with Linda.*

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

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

[5]Notice that this is trivially the case whenever $\leq$ is antisymetric, i.e. a partial order.

This inductive definition is completed by the above DeMorgan-style rewrite rules. For instance,

$$\mathsf{Ans}(X_1, m(x,y) \wedge r(x) \wedge -r(y)) = \{\langle P, L\rangle\}$$

Updates are insertions, $\mathsf{Upd}(X,a) := X \cup \{a\}$*, and deletions,* $\mathsf{Upd}(X,-a) := X - \{a\}$*. For* $E \subseteq \mathrm{At} \cup \overline{\mathrm{At}}$*, we have* $\mathsf{Upd}(X,E) = X \cup (E^+ - E^-) - (E^- - E^+)$*. For instance,*

$$\mathsf{Upd}(X_1, -m(P,L) \wedge m(P,S)) = \{r(S), r(P),\, m(P,S)\}$$

describes a possible transaction. The KS of relational databases, denoted by $\boldsymbol{A}^-$*, is then defined as*

$$\langle \emptyset,\, \subseteq,\, 2^{\mathrm{At}},\, \vdash,\, L(-,\wedge,\vee),\, \mathsf{Upd},\, \mathrm{At} \cup \overline{\mathrm{At}}\,\rangle$$

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

Example 2 (Fact Bases) *A KB consisting of ground literals (viewed as positive and negative facts) is called a* fact base*. For instance, the fact base* $X = \{\sim b(S), m(P,L)\}$ *may represent the information that Susan is not blonde, and that Peter is married with Linda.*

As a kind of natural deduction from positive and negative facts an inference relation $\vdash$ *between a fact base* $X \subseteq \mathrm{Lit}$ *and a ground formula* $F \in L(\sim,\wedge,\vee)$ *is defined in the following way:*[6]

$$\begin{array}{lrl}
(\vdash a) & X \vdash a & \textit{if} \quad a \in X \\
(\vdash \sim a) & X \vdash \sim a & \textit{if} \quad \sim a \in X \\
(\vdash \wedge) & X \vdash F \wedge G & \textit{if} \quad X \vdash F \;\;\&\;\; X \vdash G \\
(\vdash \vee) & X \vdash F \vee G & \textit{if} \quad X \vdash F \textit{ or } X \vdash G
\end{array}$$

This inductive definition is completed by the DeMorgan-style rewrite rules presented above.

Updating by a literal is defined as insertion: $\mathsf{Upd}(X,l) := X \cup \{l\}$*. The system of fact bases is then defined as*

$$\boldsymbol{F}^+ \;:=\; \langle\, \emptyset,\, \subseteq,\, 2^{\mathrm{Lit}},\, \vdash,\, L(\sim,\wedge,\vee),\, \mathsf{Upd},\, \mathrm{Lit}\,\rangle$$

If the query language does also allow for weak negation, i.e. if $L_{\mathrm{Query}} = L(-,\sim,\wedge,\vee)$*, and*

$$(\vdash -l) \qquad X \vdash -l \textit{ if } l \notin X$$

the corresponding KS will be denoted by $\boldsymbol{F}$*.*

In a more general setting, Upd *can also be defined for weakly negated literals, namely as deletion:* $\mathsf{Upd}(X,-l) := X - \{l\}$*, resp. for sets (corresponding to conjunctions) of extended literals,* $E \subseteq \mathrm{XLit}$*, as*

$$\mathsf{Upd}(X,E) := X \cup (E^+ - E^-) - (E^- - E^+)$$

The corresponding extension of $\boldsymbol{F}$ *admitting extended literals as inputs,* $L_{\mathrm{Input}} = \mathrm{XLit}$*, will be denoted by* $\boldsymbol{F}^-$*. For example, if we learn that Peter gets divorced from Linda and marries Susan, we perform the following update:*

$$\mathsf{Upd}(\{\sim b(S), m(P,L)\},\; -m(P,L) \wedge m(P,S)) = \{\sim b(S), m(P,S)\}$$

Such a sequence of basic inputs (insertions and deletions of atoms or literals) is also called a knowledge base transaction.

[6]Notice that according to $(\vdash a)$ and $(\vdash \sim a)$, both an atom and its complement are acceptable at the same time, and independently from each other.

Proof: We have to show that $\boldsymbol{F}^+$, $\boldsymbol{F}$ and $\boldsymbol{F}^-$ satisfy KS1–KS3. KS1 ist satisfied by stipulation. It is easy to check that for any extended literal $e \in \mathrm{XLit}$, $\mathsf{Upd}(X,e) \vdash e$. Hence, KS2 holds. Finally, KS3 holds trivially since the information ordering $\subseteq$ is a partial order. □

Standard logics, such as classical or intuitionistic logic, can be viewed as special (very idealized) cases of KSs. Here, the KB consists of a (possibly infinite) set of arbitrary first order sentences, and is usually called a 'theory'. The inference operation is monotonic. Updates are simple additions of formulas (corresponding to the extension of a set of premise formulas by adding a further premise formula separated by a comma), and the representation language is equal to the query and input language, $L_{\mathrm{Repr}} = L_{\mathrm{Query}} = L_{\mathrm{Input}}$.

Observation 1 (Exclusive Disjunction) *In addition to a standard disjunction* $\vee$, *an* exclusive *disjunction* $|$ *can be defined in* $\boldsymbol{F}$*:*

$$F \mid G \quad := \quad (F \wedge -G) \vee (G \wedge -F)$$

For instance, $\{p,q\} \vdash p \vee q$*, but* $\{p,q\} \not\vdash p \mid q$*.*

2.2 Some Formal Properties of KSs

The following is a list of some fundamental properties a KS may have.

2.2.1 Basic Properties of Inference

Unique Representation

$$C(\mathrm{KB}_1) = C(\mathrm{KB}_2) \;\Rightarrow\; \mathrm{KB}_1 = \mathrm{KB}_2$$

A standard logical system, where a KB is a set of well-formed formulas, does not enjoy the Unique Representation property. On the other hand, a relational database, for instance, is a unique representation.

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 1 (Persistent Queries) *A closed query formula* F *is called* persistent *if* $\mathrm{KB}_1 \vdash F \;\Rightarrow\; \mathrm{KB}_2 \vdash F$*, whenever* $\mathrm{KB}_1 \le \mathrm{KB}_2$*. If all* $F \in L_{\mathrm{Query}}$ *are persistent, the KS and its inference relation* $\vdash$ *are called persistent. An operator of the query language is called persistent, if every query formed with it and with persistent subformulas is again persistent.*

Weak negation, as an operator of the query language in $\boldsymbol{F}$, as well as the exclusive disjunction $|$, are not persistent while, e.g., queries $F \in L(\sim,\wedge,\vee)$ are persistent in $\boldsymbol{F}$.

Constructive Inference Let $L \subseteq \mathrm{Lit}$ be an arbitrary set of ground literals, and $\lceil L \rceil := \mathsf{Upd}(0, L)$. Then $\vdash$ is called *constructive* if it satisfies (i) constructible truth, and (ii) constructible falsity, i.e. both of the following conditions,

$$(i) \quad \lceil L \rceil \vdash F \vee G \;\Rightarrow\; \lceil L \rceil \vdash F \;\text{ or }\; \lceil L \rceil \vdash G$$

$$(ii) \quad \lceil L \rceil \vdash \sim (F \wedge G) \;\Rightarrow\; \lceil L \rceil \vdash \sim F \;\text{ or }\; \lceil L \rceil \vdash \sim G$$

Obviously, the first condition (of constructible truth) excludes the possibility of certain disjunctive tautologies such as the *tertium non datur*, $F \vee \sim F$, whereas its negative counterpart excludes, for instance, the *principle of contradiction*, $\sim (F \wedge \sim F)$.

The next property (due to Urbas [1990]) excludes the possibility of trivial inferences, i.e. non-tautological inferences which are solely based on the form of a query and not on its content. $F \in L_{\text{Query}}$ is called a *non-tautology* if $\text{KB} \not\vdash F$ for some $\text{KB} \in L_{\text{KB}}$.

Non-Explosive Inference $\vdash$ is called *non-explosive* if for every non-tautology $F \in L_{\text{Query}}$, and for every $\text{KB} > 0$, there is a variant F' of F (obtained by uniform substitution of propositional constituents) such that $\text{KB} \not\vdash F'$.

2.2.2 Basic Properties of Update

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

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

Permutation

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

Both Contraction and Permutation follow from the property of

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

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

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

Definition 2 (Ampliative Inputs) *An input formula F is called (i)* ampliative[7] *if* $\text{KB} \leq \mathsf{Upd}(\text{KB}, F)$*, or (ii)* reductive *if* $\text{KB} \geq \mathsf{Upd}(\text{KB}, F)$*. A KS and its update operation* Upd *are called* ampliative*, if all inputs $F \in L_{\text{Input}}$ are ampliative.*

For instance, $\boldsymbol{F}$ is ampliative while $\boldsymbol{F}^-$ is not simply because it may be the case that $\mathsf{Upd}(X, -l) \subset X$. Ampliative updating can be viewed as the formal counterpart of persistent answering.

Input Completeness

$$\forall \text{KB}_1, \text{KB}_2 \in L_{\text{KB}} \exists F \in L_{\text{Input}} : \text{KB}_2 = \mathsf{Upd}(\text{KB}_1, F)$$

A KS is input complete iff KBs are both input constructible and input destructible, i.e. both of the following conditions hold:

$$\begin{array}{ll} (i) & \forall \text{KB} \in L_{\text{KB}} \exists F \in L_{\text{Input}} : \text{KB} = \mathsf{Upd}(0, F) \\ (ii) & \forall \text{KB} \in L_{\text{KB}} \exists F \in L_{\text{Input}} : \mathsf{Upd}(\text{KB}, F) = 0 \end{array}$$

[7] The name is adopted from [Belnap 1977].

2.2.3 Properties of Interaction

Lemma Redundancy (alias: Cut, Transitivity)

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

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

2.2.4 Vivid Knowledge Representation and Reasoning

In a KS with a sophisticated inconsistency handling mechanism, it should be impossible that both a query F and its negation can ever be inferred from a KB even if the KB contains contradictory information. Let $\neg$ be a negation operator.

Inherent Consistency $\quad \mathrm{KB} \vdash \neg F \;\Rightarrow\; \mathrm{KB} \not\vdash F$

Notice that this does not hold for classical negation. In fact, Inherent Consistency is violated by any negation satisfying the classical principle *ex contradictione sequitur quodlibet*, $\{F, \sim F\} \vdash G$.

Minimal Change for Literal Expansion $\mathrm{KB} + l := \mathsf{Upd}(\mathrm{KB}, l)$ is the least extension of KB such that l can be inferred, expressed by (+1) and (+2):

$$\begin{array}{ll} (+1) & \mathrm{KB} + l \geq \mathrm{KB} \\ (+2) & \mathrm{KB}' \geq \mathrm{KB} \;\;\&\;\; \mathrm{KB}' \vdash l \;\Rightarrow\; \mathrm{KB} + l \leq \mathrm{KB}' \end{array}$$

Definition 3 (Basic VKS) *A KS with standard operators $\sim$, $\wedge$ and $\vee$, satisfying Constructive and Non-Explosive Inference, and the principle of Minimal Change for Literal Expansion is called a* basic *vivid knowledge system* (VKS).

Concerning the weak negation '$-$' (alias negation-as-failure) it is not sufficient, in general, that F fails in order to infer $-F$ (it is sufficient, however, in the case of definite KBs). Rather, it is necessary, i.e. weak negation satisfies Inherent Consistency: $\mathrm{KB} \vdash -F \;\Rightarrow\; \mathrm{KB} \not\vdash F$. If weakly negated inputs are allowed, the following postulate should be satisfied:

Minimal Change for Literal Contraction $\mathrm{KB} - l := \mathsf{Upd}(\mathrm{KB}, -l)$ is the greatest reduction of KB such that $-l$ can be inferred, expressed by (-1) and (-2):

$$\begin{array}{ll} (-1) & \mathrm{KB} - l \leq \mathrm{KB} \\ (-2) & \mathrm{KB}' \leq \mathrm{KB} \;\;\&\;\; \mathrm{KB}' \vdash -l \;\Rightarrow\; \mathrm{KB} - l \geq \mathrm{KB}' \end{array}$$

Definition 4 (VKS) *A VKS is a basic VKS extended by adding a weak negation operator '$-$' such that Inherent Consistency and Minimal Change for Literal Contraction (if weak negation is admitted in the input language) hold, and the following* Closed-World *reasoning principle is stipulated:*

$$(CWA) \quad \langle CWA, \mathrm{KB}\rangle \vdash \sim p(t) \quad \textit{if} \quad p \in CWA \;\;\&\;\; \mathrm{KB} \vdash -p(t)$$

where CWA denotes the set of those predicate symbols of a given language which are subject to the Closed-World Assumption (i.e. which are totally represented in the knowledge base).[8]

Alternatively, we can also define inference from $\langle \mathrm{CWA}, \mathrm{KB}\rangle$ by means of the *CWA completion* of KB:

$$CWA(\mathrm{KB}) := \mathsf{Upd}(\mathrm{KB}, \{\sim p(t) : p \in \mathrm{CWA} \;\;\&\;\; \mathrm{KB} \vdash -p(t))$$

Obviously, we have $\langle \mathrm{CWA}, \mathrm{KB}\rangle \vdash F$ iff $\langle \emptyset, CWA(\mathrm{KB})\rangle \vdash F$, resp. $CWA(\mathrm{KB}) \vdash F$.

Although we do not make it a strict requirement, it seems desirable for a VKS that the following *coherence*[9] property, relating weak with strong negation, holds:

Negation Coherence If KB is consistent, then $\mathrm{KB} \vdash \sim F \;\Rightarrow\; \mathrm{KB} \vdash -F$

Using both kinds of negation one can express a vivid form of the well-known *Rational Monotonicity* postulate,[10] namely

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

which is a two-fold weakening of the rather unrealistic original one,

$$\mathrm{KB} \vdash F \;\;\&\;\; \mathsf{Upd}(\mathrm{KB}, G) \not\vdash F \;\Rightarrow\; \mathrm{KB} \vdash \sim G$$

2.3 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.[11]

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

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

There are two 'parameters' on which Monotonicity depends: the update operation must be ampliative, and the inference relation must be persistent. If either of these conditions is violated, the resp. KS is nonmonotonic.

Observation 2 *A KS is monotonic iff it is ampliative and persistent (even if its update operation is nonmonotonic).*

---

[8] The Closed-World Assumption, in a less general form, was originally proposed in [Rei78]. Notice that our form of the CWA relates explicit with default-implicit falsity, i.e. strong with weak negation: an atomic sentence formed with a totally represented predicate is (explicitly) false if it is false by default, i.e. its strong negation holds if its weak negation does. A dual version for establishing the truth of an atom via the CWA can be added, as discussed in [Wag94a].

[9] The name is adopted from [PA92].

[10] This was brought to my attention by David Pearce.

[11] Notice that Monotonicity implies Lemma Compatibility.

For instance, $\boldsymbol{A}$ is nonmonotonic since it allows for non-persistent queries. On the other hand, $D\boldsymbol{F}^{+}$, admitting only for ampliative inputs and persistent queries, $L_{\mathrm{Query}} = \mathrm{Lit}^{\wedge}$, is monotonic even though its update operation, by virtue of the inconsistency handling principle mi, is nonmonotonic.

Chapter 3

Disjunctive Knowledge Bases

Using the general concept of a *knowledge representation and reasoning system* proposed in [Wag94a] we show how Belnap's system of *epistemic states* can be extended by adding a second negation in the style of Negation-as-Failure. We call the resulting nonmonotonic system with two kinds of negation *Disjunctive Fact Bases*. We also formalize Körner's distinction between exact and inexact predicates for epistemic states and disjunctive fact bases, and show how to obtain classical logic by declaring all predicates as exact. In the system of disjunctive fact bases, it is possible to simplify the representation of exact predicates (and reduce the complexity created by them) by means of an appropriate form of Reiter's Closed-World Assumption (CWA). Thus, we finally distinguish between three kinds of predicates: inexact predicates, exact predicates subject to the CWA, and exact predicates not subject to the CWA.

As opposed to Belnap's epistemic states, in disjunctive fact bases one can distinguish between exclusive and inclusive disjunctive information, and express various kinds of incomplete information which can only be queried by means of appropriate epistemic modal operators. We therefore extend the query language of $D\mathbf{F}$ by introducing an S5-style modal operator for definite knowledge.

3.1 Introduction

In this paper, we shall address issues of disjunctive information processing, resp. disjunctive knowledge representation and reasoning, such as the interplay between disjunctive and negative information (concerning negation-as-failure and the Closed-World Assumption), and the notion of inclusive versus exclusive disjunctive information. These issues have first been discussed in the field of disjunctive logic programming (see, e.g. [Min82, RT88, GL91, LMR92]). However, there is no generally acknowledged semantics for disjunctive logic programs, and many proposals are not based on clear logical principles but seem to be rather ad-hoc. It appears that the non-rule-related issues of disjunctive information processing are complicated by the presence of deduction rules, and a semantical account for the interplay between disjunctive, negative and deductive knowledge is hard to find. Therefore, we believe it is methodologically preferable to settle these issues in a simpler framework without rules, and this is the approach taken here.

The concept of a knowledge representation and reasoning system, or shorter: *knowledge system (KS)*, consists essentially of two main components: an inference and an update operation manipulating knowledge bases as abstract objects,[1] together with a set of formal properties these operations may have. In general, there are no specific restrictions on the

[1] This with respect to KR fundamental distinction was first proposed in [Lev84] where the resp. operations are called *ASK* and *TELL*.

internal structure of a knowledge base. It appears, however, that a computational design can be achieved by 'compiling' incoming information into some normal form rather than leaving it in the form of arbitrarily complex formulas. This is the case, for instance, in Belnap's KS which can be considered as the paradigm for KSs.

The concept of a KS constitutes a useful framework for the classification and comparison of various computational systems and formalisms like, e.g., relational and deductive databases, logic programs and other rule-based systems. It is more general than that of a logic (i.e. a consequence relation). A standard logic can be viewed as a special kind of KS. On the other hand, by defining the inference and update operations procedurally, KSs can serve as the basis for the operational definition of logics. For instance, by appropriate settings of certain parameters in the system of disjunctive fact bases one can obtain three-valued and classical logic, in addition to Belnap's four-valued logic.

Both relational and deductive databases can be considered as computational paradigms of real world knowledge systems. They implement a form of nonmonotonic reasoning caused by the use of negation-as-failure refering to default-implicit negative information. On the other hand, relational and deductive databases, as well as normal logic programs, are not capable of representing and processing explicit negative information. This shortcoming has led to the extension of logic programming by adding a second negation (in addition to negation-as-failure) as proposed independently in [GL90, GL91] and in [PW90, Wag91a]. We call the general concept of an operator expressing default-implicit negative information in the style of negation-as-failure *weak negation*, and denote it by '$-$', while the concept of an operator expressing explicit negative information will be called *strong negation*, denoted by '$\sim$'. Our concept of a *vivid knowledge system (VKS)* is a two-fold generalization:

1. it extends already known logics, such as Belnap's 4-valued or Nelson's constructive logic,[2] by adding *weak* negation, and

2. it extends already known knowledge systems, such as relational or deductive database systems, by adding *strong* negation.

In the framework of a VKS, a specific meaning is assigned to the *Closed-World Assumption*: if the Closed-World Assumption holds for a predicate, its weak negation implies its strong negation, in other words, an atomic sentence formed with such a predicate is already false if it is false by default.

While negation-as-failure in logic programming is only defined as an operator of the query language, i.e. on the right hand side of the turnstile, we shall propose a kind of elimination rule for negation-as-failure, thus admitting it as well on the left hand side of the turnstile, resp. as an operator of the input language. The elimination rule is defined by means of the contraction $\mathrm{KB} - F$ of KB by F:

$$\frac{\mathrm{KB} - F \vdash G}{\mathrm{KB}, -F \vdash G}$$

The structure of the paper is as follows: in section 2 we shall present some basic concepts of the theory of knowledge representation proposed in [Wag94a], in section 3 we formulate Belnap's epistemic states in terms of this theory, in section 4 the system of disjunctive fact bases is defined, and in section 5 the distinction between exact and inexact predcates and the Closed-World Assumption are formalized.

[2]See [Bel77], resp. [AN84].

3.2 Belnap's Epistemic States

A KS, according to Belnap, answers queries by invoking some inference mechanism, and it accepts input from a variety of sources by using an appropriate update mechanism. In such circumstances inconsistency threatens. Mr. X tells the KS that A while Mrs. Y tells it that $\sim A$. Or, in a different environment, an automatic measurement yields that $m > 0.3$ while the subsequent confirmation attempt yields that $m < 0.3$.

What is the KS to do ?

Possibility 1: Refuse to accept inconsistent information. However: this is unfair either to Mr. X or to Mrs. Y. Also, contradictions may not lie on the surface.

Possibility 2: Revise current beliefs in the presence of contradictions. However: it seems to be difficult to determine the proper revision policy doing justice to both Mr. X and Mrs. Y, and it seems to be even more difficult to mechanize it in a satisfactory way.

Possibility 3: Just accept contradictions and report them exactly as they were told, so the user can make up her mind.

Belnap advocates possibility 3, but emphasizes that even if the ultimate goal is possibility 2, i.e. revision, possibilty 3 is a good first step towards that goal.

In order to be able to process arbitrary input formulas $F \in L(\sim, \wedge, \vee)$, an epistemic state has to consist of a set of fact bases (corresponding to partial interpretations, called 'set-ups' by Belnap), i.e. an epistemic state Y is a finite subset of 2^{Lit}, the elements of Y representing different epistemic alternatives. Following Belnap, yet formulated differently, inputs are processed as follows:

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

All other cases of compound formulas are treated by DeMorgan-like rewriting (see the rules in chapter 2).

There are two other update operations differing from Upd_B with respect to inconsistency handling.

$$\begin{aligned} \mathsf{Upd}_{ex}(Y, F) &:= \{X \in \mathsf{Upd}_B(Y, F) : X \cap \widetilde{X} = \emptyset\} \\ \mathsf{Upd}_{mi}(Y, F) &:= \{X \in \mathsf{Upd}_B(Y, F) : \neg\exists Z \in \mathsf{Upd}_B(Y, F), \text{ such that} \\ &\qquad (Z \cap \widetilde{Z}) \subset (X \cap \widetilde{X})\} \end{aligned}$$

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}.

A query formula $F \in L(\sim, \wedge, \vee)$ can be inferred from an epstemic state Y if it can be inferred from every fact base $X \in Y$:

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

where $X \vdash F$ is the inference relation of $\boldsymbol{F}^+$ (see ex. 1).

The process of information growth can be captured by the following notion of informational extension. An epistemic state Y' is called an *(informational) extension* of Y, symbolically $Y \leq Y'$, if every epistemic alternative in Y' extends one in Y:

Definition 5 (Information Ordering) $Y \leq Y' :\Longleftrightarrow \forall X' \in Y' \, \exists X \in Y : X \subseteq X'$

This ordering, used in [Bel77], was also suggested in domain theory for the semantics of parallel processes (see [Plo76, Smy78]), and is sometimes called 'Smyth ordering'.

Observation 3 *$\langle 2^{2^{\mathrm{Lit}}}, \leq \rangle$ is a preorder with least element $0 := \{\emptyset\}$.*

Notice that while $\{\mathrm{Lit}\}$ is informationally larger than any non-empty epistemic state $Y \subseteq 2^{\mathrm{Lit}}$, the empty epistemic state is still larger:

$$Y \leq \{\mathrm{Lit}\} < \{\}$$

Let $\min Y = \{X \in Y : \neg \exists X' \in Y (X' \subset X)\}$ be the restriction of Y to its minimal elements. An epistemic state Y is called *minimal* if $Y = \min Y$. The collection of all minimal epistemic states forms a lattice, as shown in [KM93].

Observation 4 *$\langle \{Y \subseteq 2^{\mathrm{Lit}} : Y = \min Y\}, \leq \rangle$ is a lattice order. Meet and join can be defined by $Y_1 \sqcap Y_2 := \min(Y_1 \cup Y_2)$, and $Y_1 \sqcup Y_2 := \min\{X_1 \cup X_2 : X_1 \in Y_1, X_2 \in Y_2\}$.*

Definition 6 (Belnap's KS)

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

is called Belnap's KS.

We have to show that KS1–KS3 are satisfied.

KS1: Notice that obviously $Y \vdash 1$, and by stipulation, $\mathsf{Upd}(Y, 1) = Y$.

KS2: Reflexivity is proved by induction on input formulas. Clearly, $\mathsf{Upd}(Y, l) \vdash l$, since l is added to every element of Y. For conjunctive inputs we obtain $\mathsf{Upd}(Y, F \wedge G) = (Y + F) + G \vdash G$, by the induction hypothesis, and since $\boldsymbol{B}$ is monotonic it follows from $Y + F \vdash F$ that $(Y + F) + G \vdash F$, and hence $(Y + F) + G \vdash F \wedge G$. For disjunctive inputs, $\mathsf{Upd}(Y, F \vee G) = (Y + F) \cup (Y + G) \vdash F \vee G$, since for every $X \in (Y + F)$, it holds by the induction hypothesis that $X \vdash F$, hence $X \vdash F \vee G$, and similarly for every $X \in (Y + G)$.

KS3: It suffices to show that $Y_1 \leq\geq Y_2$ iff $C(Y_1) = C(Y_2)$. This follows from the facts that 1) $Y_1 \leq\geq Y_2$ iff both have the same minimal elements, i.e. iff $\min Y_1 = \min Y_2$, and 2) $C(Y) = C(\min Y)$, as shown in [KM93]. □

Claim 1 *Belnap's KS satisfies Monotonicity, i.e. $C(Y) \subseteq C(\mathsf{Upd}(Y, F))$.*

Proof: see [Wag94a].

Observation 5 *In $\boldsymbol{B}$, $\vdash$ is persistent, i.e. if $Y' \geq Y$ then $Y' \vdash F$ whenever $Y \vdash F$ for all $F \in L(\sim, \wedge, \vee)$.*

When an update takes place, the current information never decreases, in other words, the update operation of $\boldsymbol{B}$ is *ampliative*:

Observation 6 $Y \leq \mathsf{Upd}_B(Y,F) \leq \mathsf{Upd}_{mi}(Y,F) \leq \mathsf{Upd}_{ex}(Y,F)$

Let $\mathrm{DNS}(F)$ denote the *disjunctive normal set* corresponding to the disjunctive normal form of F.

Observation 7

$$\mathsf{Upd}(Y,F) = \{X \cup K : X \in Y, K \in \mathrm{DNS}(F)\}$$

Proof: See [Wag94a]. This observation implies that Contraction and Permutation hold in $\boldsymbol{B}$.

Claim 2 (Four-Valued Logic) *Let $\models_4 \subseteq L^2(\sim,\wedge,\vee)$ denote the consequence relation determined by general partial (resp. four-valued) models wrt standard truth conditions (resp. tables)*[3]*, then in $\boldsymbol{B}$*

$$F \models_4 G \quad \textit{iff} \quad \mathsf{Upd}(0,F) \vdash G$$

Proof: From the previous observation it follows that $\mathsf{Upd}(0,F) = \mathrm{DNS}(F)$. A further observation from [Wag94a] says that $\mathrm{DNS}(F) = \mathrm{MinMod}_4(F)$. Hence, $\mathsf{Upd}(0,F) = \mathrm{MinMod}_4(F)$. This, together with the facts that 1) propositional entailment in partial logic is determined by minimal models $M \subseteq \mathrm{Lit}$, and 2) for every $M \in \mathrm{MinMod}_4(F)$, $M \models_4 G$ iff $\mathsf{Upd}(0,M) = \{M\} \vdash G$, yields the assertion. □

3.2.1 Adding an S5-like Modal Operator for Definite Belief

The query language of $\boldsymbol{B}$ can be extended by adding the modal operator B for definite belief, yielding the *S5 query extension* $\boldsymbol{B}^5$. For $X \in Y \subseteq 2^{\mathrm{Lit}}$, we define:[4]

(B)	$X \vdash \mathsf{B}F$	if	for all $X' \in Y : X' \vdash F$
$(\sim\mathsf{B})$	$X \vdash \sim\mathsf{B}F$	if	for some $X' \in Y : X' \vdash \sim F$

A corresponding 'possibility' operator is definable:

$$\mathsf{E}F := \sim\mathsf{B}\sim F$$

expressing that there is some evidence in favour of F. Thus we get the following system:

$$\boldsymbol{B}^5 \;:=\; \langle\, \{\emptyset\},\, \leq,\, \mathrm{Fin}(2^{2^{\mathrm{Lit}}}),\, \vdash,\, L(\sim,\wedge,\vee,\mathsf{B},\mathsf{E}),\, \mathsf{Upd}_B,\, L(\sim,\wedge,\vee)\,\rangle$$

Observation 8 (Definite vs. Indefinite Existence) *By means of B, indefinite existential queries can be distinguished from definite ones. For instance, if*

$$X = \{\{p(c), q(d)\},\ \{p(d), q(d)\}\}$$

we obtain $X \vdash \exists x p(x)$ as well as $X \vdash \exists x q(x)$, but only $X \vdash \exists x\, \mathsf{B}q(x)$, whereas $X \not\vdash \exists x\, \mathsf{B}p(x)$. Gelfond [1992] proposes to use such existential queries as integrity constraints requiring that a KB knows definitely, or indefinitely, certain entities satisfying certain predicates.

---

[3] See [Wag94a].

[4] Strictly speaking, we should rather write $Y, X \vdash F$, specifying the set of 'possible worlds', Y, along with the 'actual world' X (as usual in possible worlds semantics), instead of $X \vdash F$. We prefer, however, the more economic notation, keeping the context Y in the background whenever it is clear.

3.2.2 Problems with Belnap's KS

We briefly discuss three problems with $\boldsymbol{B}$.

No Unique Representation

In $\boldsymbol{B}$, epistemic states are not unique representations: $C(Y) = C(Y')$ does not imply that $Y = Y'$. For instance,

$$C(\{\{p\}\}) = C(\{\{p\}, \{p,q\}\})$$

Only the minimal elements of an epistemic state count:

$$C(Y) = \bigcap\{C(X) : X \in Y\} = \bigcap\{C(X) : X \in \min Y\}$$

Therefore, exclusive and inclusive disjunctive information cannot be distinguished, neither by comparing the resp. information content nor by being appropriately queried. For instance,

$$\begin{array}{rcl} \{\{p\}, \{q\}\} & \leq\geq & \{\{p\}, \{q\}, \{p,q\}\} \\ C(\{\{p\}, \{q\}\}) & = & C(\{\{p\}, \{q\}, \{p,q\}\}) \end{array}$$

No Disjunctive Syllogism

For instance,

$$\mathsf{Upd}_B(0, (p \vee q) \wedge {\sim}p) = \{\{p, {\sim}p\}, \{q, {\sim}p\}\} \not\vdash q$$

Notice that the resulting epistemic state contains an inconsistent epistemic alternative although the input itself was consistent.

The Disjunctive syllogism holds in $\boldsymbol{B}_{ex}$, where all inconsistent epistemic alternatives are discarded. In this system, however, if $Y \vdash l$ then $\mathsf{Upd}(Y, \tilde{l}) = \{\}$ yields an 'exploded' KB in the sense that everything follows from $\{\}$, i.e. $\boldsymbol{B}_{ex}$ is explosive. A good compromise seems to be $\boldsymbol{B}_{mi}$ which is both non-explosive and satisfies the Disjunctive Syllogism.

No Input Completeness

Finite epistemic states are input constructible in $\boldsymbol{B}$, i.e.

$$\forall Y \in \mathrm{Fin}(2^{\mathrm{Fin}(2^{\mathrm{Lit}})}) \exists F \in L_{\mathrm{Input}} : Y = \mathsf{Upd}(0, F)$$

but since they are not input destructible $\boldsymbol{B}$ does not satisfy Input Completeness.

3.3 Disjunctive Databases

A Disjunctive Database is a finite collection of sets of atoms (each one describing a *possible situation*). For instance, if we know that Susan is a resident and that Peter is married either with Linda or with Susan, we get the following disjunctive DB:

$$Y_1 = \{\{r(S), m(P,L)\}, \{r(S), m(P,S)\}\}$$

Formally, the system of Disjunctive Databases is defined as:[5]

$$D\boldsymbol{A}^- := \langle\, \{\emptyset\}, \leq, \mathrm{Fin}(2^{2^{\mathrm{At}}}), \vdash, L(-, \wedge, \vee, |), \mathsf{Upd}, L^*(-, \wedge, \vee)\,\rangle$$

[5] We shall use the following notation: if K^- is a KS allowing for weakly negated input, K is the corresponding without weak negation in the input language, and in K^+ weak negation is also dropped from the query language.

where the input language $L^*(-,\wedge,\vee)$ excludes those formulas where an atom occurs both in the scope of an odd and an even number of negations (i.e. potentially inconsistent inputs).

The information ordering between disjunctive databases $Y_1, Y_2 \subseteq 2^{\mathrm{At}}$ is defined as

$$\begin{array}{lll} Y_1 \leq Y_2 & :\Longleftrightarrow & \forall X_2 \in Y_2 \exists X_1 \in Y_1 : X_1 \subseteq X_2, \text{ and} \\ & & \text{either (i)} \quad Y_1 \subseteq Y_2, \\ & & \text{or (ii)} \quad \exists X_1' \in Y_1 \forall X_2 \in Y_2 : X_1' \not\supseteq X_2 \end{array}$$

According to this definition, we obtain for instance,

$$\begin{array}{rcl} \{\{p\}\} & \leq & \{\{p\},\{p,q\}\}, \text{ but } \{\{p\}\} \not\geq \{\{p\},\{p,q\}\} \\ \{\{p\}\} & \geq & \{\{p\},\{p,q\},\{q\}\} \\ \{\{p\},\{q\}\} & \leq & \{\{p\},\{p,q\},\{q\}\}, \text{ but } \{\{p\},\{q\}\} \not\geq \{\{p\},\{p,q\},\{q\}\} \end{array}$$

For $Y \subseteq 2^{\mathrm{At}}$,

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

where $X \vdash F$ is defined in the same way as in $\boldsymbol{A}$. Inputs are processed as follows:

$$\begin{array}{llcl} (Ua) & \mathsf{Upd}(Y,a) & = & \{X \cup \{a\} : X \in Y\} \\ (U{-}a) & \mathsf{Upd}(Y,-a) & = & \{X - \{a\} : X \in Y\} \\ (U\wedge) & \mathsf{Upd}(Y, F \wedge G) & = & \mathsf{Upd}(\mathsf{Upd}(Y,F),G) \\ (U\vee) & \mathsf{Upd}(Y, F \vee G) & = & \mathsf{Upd}(Y,F) \cup \mathsf{Upd}(Y,G) \cup \mathsf{Upd}(Y, F \wedge G) \end{array}$$

For instance, if we have Y_1, and we then learn that Peter or Linda is a resident, we perform the following update:

$$\begin{array}{rcl} Y_2 & := & \mathsf{Upd}(Y_1, r(P) \vee r(L)) \\ & = & \{\{r(P), r(S), m(P,L)\}, \{r(P), r(S), m(P,S)\}, \\ & & \{r(L), r(S), m(P,L)\}, \{r(L), r(S), m(P,S)\}, \\ & & \{r(P), r(L), r(S), m(P,L)\}, \{r(P), r(L), r(S), m(P,S)\}\} \end{array}$$

As can be easily checked, there is no married couple such that not both are residents:

$$\mathsf{Ans}(Y_2, m(x,y) \wedge -(r(x) \wedge r(y))) = \emptyset$$

There are two answers to the query "who is a resident or married with a resident ?",

$$\mathsf{Ans}(Y_2, r(x) \vee \exists y(m(x,y) \wedge r(y))) = \{S, P\}$$

although one can neither infer that Peter is a resident nor that he is married with one.

Notice that compared to $\boldsymbol{B}$ we have changed three things in $D\boldsymbol{A}$: 1) we have weak negation (and, by definition, exclusive disjunction) in the query language, 2) we have refined the information ordering, and 3) we have modified the update operation for disjunctive inputs in order to give them an inclusive meaning.

Observation 9 $\langle 2^{2^{\mathrm{At}}}, \leq \rangle$ *is a partial order with least element* $0 := \{\emptyset\}$.

Proof: (Transitivity) We have to show that $Y_1 \leq Y_2$ and $Y_2 \leq Y_3$ implies $Y_1 \leq Y_3$. We write $Y_1 \leq' Y_2$ instead of $\forall X_2 \in Y_2 \exists X_1 \in Y_1 : X_1 \leq X_2$. Notice that $\leq'$ is transitive, and that from $x_2 \not\leq x_1$ and $x_2 \leq x_3$, we can conclude for any transitive relation $\leq$ that $x_3 \not\leq x_1$. Assuming that $Y_1 \leq Y_2$ and $Y_2 \leq Y_3$, implying $Y_1 \leq' Y_2 \leq' Y_3$, we have to consider three cases.

Case 1: $Y_1 \subseteq Y_2 \subseteq Y_3$. Then obviously $Y_1 \subseteq Y_3$, and hence $Y_1 \leq Y_3$.

Case 2: $\exists X_1' \in Y_1 \forall X_2 \in Y_2 : X_2 \not\leq X_1'$. Since $\forall X_3 \in Y_3 \exists X_2 \in Y_2 : X_2 \leq X_3$, we obtain $\forall X_3 \in Y_3 : X_3 \not\leq X_1'$, and consequently, $Y_1 \leq Y_3$.

Case 3: $\exists X_2' \in Y_2 \forall X_3 \in Y_3 : X_3 \not\leq X_2'$. Since $\forall X_2 \in Y_2 \exists X_1 \in Y_1 : X_1 \leq X_2$, there is $X_1' \in Y_1$, such that $X_1' \leq X_2'$, and consequently $\exists X_1' \in Y_1 \forall X_3 \in Y_3 : X_3 \not\leq X_1'$, i.e. $Y_1 \leq Y_3$.

(Antisymmetry) It is easy to see that $Y_1 \leq Y_2 \leq Y_1$ implies that $Y_1 \subseteq Y_2 \subseteq Y_1$, and hence $Y_1 = Y_2$. □

3.3.1 Inclusive and Exclusive Disjunctive Information

Notice that $(U\vee)$ expresses inclusive disjunction in the following sense:

$$\mathsf{Upd}(0, p \vee q) = \{\{p\}, \{q\}, \{p, q\}\} \not\vdash -p \vee -q$$

Clearly, this corresponds to the standard meaning of a disjunction, expressing all three possibilities how it can be satisfied. By contrast, exclusive disjunction corresponds to

$$\mathsf{Upd}(0, p|q) = \{\{p\}, \{q\}\} \vdash -p \vee -q$$

In $D\boldsymbol{A}$, an inclusive disjunction contains more information than the corresponding exclusive one:

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

3.3.2 Adding an S5-like Modal Operator for Definite Belief

The query language of $D\boldsymbol{A}$ can be extended by adding the modal operator B for definite belief, yielding the *S5 query extension* $D^5\boldsymbol{A}$. For $X \in Y \subseteq 2^{\mathrm{At}}$, we define:

$$\begin{array}{llll} (\mathsf{B}) & X \vdash \mathsf{B}F & \text{if} & \text{for all } X' \in Y : X' \vdash F \\ (-\mathsf{B}) & X \vdash -\mathsf{B}F & \text{if} & \text{for some } X' \in Y : X' \vdash -F \end{array}$$

A corresponding 'possibility' operator is definable:

$$\mathsf{E}F := -\mathsf{B} - F$$

expressing that there is some evidence in favour of F. This yields

$$D^5\boldsymbol{A}^- \ := \ \langle\, \{\emptyset\},\, \leq,\, \mathrm{Fin}(2^{2^{\mathrm{At}}}),\, \vdash,\, L(-, \wedge, \vee, |, \mathsf{B}, \mathsf{E}),\, \mathsf{Upd},\, L^*(-, \wedge, \vee)\,\rangle$$

Observation 10 (Two Kinds of Failure) *Besides the* definite failure *expressed by weak negation,* $\mathrm{KB} \vdash -F$*, there is also an* epistemic failure *in* $D^5\boldsymbol{A}$ *expressed by* $\mathrm{KB} \vdash -\mathsf{B}F$*. For instance, for* $Y = \{\{p\}, \{p, q\}\}$*, we obtain* $Y \vdash -\mathsf{B}q$*, while* $Y \not\vdash -q$*.*

Observation 11 (Honest and Dishonest Disjunctive Information) *Knowing* $p \vee q$ *subsumes four cases, two of them 'honest' and two 'dishonest'. In the case of a dishonest disjunction, more information is, in fact, known than expressed by the resp. disjunction:*

- *Honest disjunctive information:*
  - *Only knowing* $p \vee q$*, and nothing more about* p *or* q*.*

$$\{\,\{p\},\, \{q\},\, \{p, q\}\,\} \vdash (p \vee q) \wedge -\mathsf{B}p \wedge -\mathsf{B}q$$

– *Knowing that either p or q but not both, and not knowing which of them:*

$$\{\{p\}, \{q\}\} \vdash (p \mid q) \wedge -\mathsf{B}p \wedge -\mathsf{B}q$$

- *Dishonest disjunctive information:*
  – *Knowing exactly one of both items: either p, or q. This corresponds to the KB* $\{\{p\}\}$, *resp.* $\{\{q\}\}$, *and to the query* $p \wedge -q$, *resp.* $q \wedge -p$.
  – *Knowing both p and q which corresponds to the KB* $\{\{p, q\}\}$, *and to the query* $p \wedge q$.

3.4 Disjunctive Fact Bases

In a similar way as we have extended the monotonic system $\boldsymbol{F}^+$ to Belnap's monotonic disjunctive KS in section 3, we now extend the nonmonotonic system $\boldsymbol{F}$ to a nonmonotonic disjunctive KS.

A *disjunctive fact base* is a finite collection of fact bases (each one describing a *possible situation*). For instance, if we know that Susan is not blonde and that Peter is married either with Linda or with Susan, we get the following KB:

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

For $Y \subseteq 2^{\mathrm{Lit}}$, inference is defined, as in $\boldsymbol{B}$, elementwise:

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

where $X \vdash F$ is inference in $\boldsymbol{F}$, i.e. $L_{\mathrm{Query}} = L(-, \sim, \wedge, \vee, |)$. For example,

$$Y_1 \vdash \sim b(S) \wedge (m(P, L) | m(P, S))$$

The information ordering between disjunctive fact bases $Y_1, Y_2 \subseteq 2^{\mathrm{Lit}}$ is defined as

$$\begin{array}{lll} Y_1 \leq Y_2 & :\Longleftrightarrow & \forall X_2 \in Y_2 \exists X_1 \in Y_1 : X_1 \subseteq X_2, \text{ and} \\ & & \text{either (i)} \quad Y_1 \subseteq Y_2, \\ & & \text{or (ii)} \quad \exists X_1' \in Y_1 \forall X_2 \in Y_2 : X_1' \not\supseteq X_2 \end{array}$$

Inputs are processed as follows:

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

We define the same additional update operations Upd_{mi}, and Upd_{ex}, as in $\boldsymbol{B}$. The only difference with respect to update in $\boldsymbol{B}$ is the definition $(U\vee)$ for disjunctive inputs which renders disjunctive information now explicitly 'inclusive'.

Example 3 *If we have the above* Y_1, *and we then learn that Susan or Linda is blonde, the following update is performed:*

$$\begin{array}{rcl} Y_2 & := & \mathsf{Upd}_{mi}(Y_1, b(S) \vee b(L)) \\ & = & \{\, \{b(L), m(P, L), \sim b(S)\},\ \{b(L), m(P, S), \sim b(S)\}\, \} \\ & \vdash & b(L) \end{array}$$

Formally, the system of disjunctive fact bases is defined as

$$D_B\boldsymbol{F}^- \ := \ \langle \{\emptyset\},\ \leq,\ \mathrm{Fin}(2^{2^{\mathrm{Lit}}}),\ \vdash,\ L(-,\sim,\wedge,\vee,|),\ \mathsf{Upd}_B,\ L^*(-,\sim,\wedge,\vee)\,\rangle$$

Correspondingly, $D_{mi}\boldsymbol{F}^-$ is formed with $L_{\mathrm{KB}} = \mathrm{mi}(\mathrm{Fin}(2^{2^{\mathrm{Lit}}}))$ and Upd_{mi}, and $D_{ex}\boldsymbol{F}^-$ with $L_{\mathrm{KB}} = \mathrm{ConsFin}(2^{2^{\mathrm{Lit}}})$ and Upd_{ex}. In the sequel, if we simply write $D\boldsymbol{F}$, resp. Upd, we mean all three systems.

The proof of KS1 and KS2 is as in the previous section for $\boldsymbol{B}$. KS3 holds trivially since the information ordering of $D_B\boldsymbol{F}$ is a partial order (see the resp. observation below).

Observation 12 *$D_B\boldsymbol{F}$ violates the principle of Coherence: if, e.g., $Y = \{\{p, \sim p\}\}$, then $Y \vdash \sim p$, but $Y \not\vdash -p$. On the other hand, Coherence holds in $D_{mi}\boldsymbol{F}$ and in $D_{ex}\boldsymbol{F}$.*

Observation 13 *$D_{mi}\boldsymbol{F}$ is nonexplosive and coherent. It has (a restricted form of) the Disjunctive Syllogism. E.g.,*

$$\begin{aligned} \mathsf{Upd}_{mi}(0, (p \vee q) \wedge \sim p) &= \{\{q, \sim p\}\} \vdash q, \textit{ while} \\ \mathsf{Upd}_{mi}(\{\{\sim q\}\}, (p \vee q) \wedge \sim p) &= \{\{p, \sim p, \sim q\},\ \{\sim p, q, \sim q\}\} \not\vdash q \end{aligned}$$

This example also shows that neither Permutation nor Update Monotonicity hold in $D_{mi}\boldsymbol{F}$.

Claim 3 (Collapse of weak and strong negation) *Let $Y \subseteq 2^{\mathrm{Lit}}$ be consistent (i.e. contain at least one consistent possible situation description). Then, whenever $Y \vdash a \vee \sim a$, weak and strong negation coincide for a in $D_{mi}\boldsymbol{F}$ and $D_{ex}\boldsymbol{F}$, but not in $D_B\boldsymbol{F}$:*

$$Y \vdash \sim a \quad \textit{iff} \quad Y \vdash -a$$

Proof: Since Y is consistent, all possible situation descriptions $X \in Y$ are consistent, and consequently, $a \vee \sim a$ is derivable from Y iff either $a \in X$ or $\sim a \in X$ for all $X \in Y$, subsuming three cases:

1. If all $X \in Y$ contain a, then neither $Y \vdash \sim a$ nor $Y \vdash -a$.
2. If all $X \in Y$ contain $\sim a$, then both $Y \vdash \sim a$ and $Y \vdash -a$.
3. Otherwise some $X \in Y$ contain a, and all others contain $\sim a$, hence $Y \not\vdash \sim a$, and $Y \not\vdash -a$. □

Claim 4 *A disjunctive fact base is a unique representation.*

Proof: See [Wag94a].

Claim 5 *$D\boldsymbol{F}$ is non-cumulative since it does not satisfy Lemma Compatibility.*

Proof by a simple counter-example:

$$\begin{array}{ll} & \{\{p\}\} \vdash p \vee q \\ \text{but} & \mathsf{Upd}(\{\{p\}\}, p \vee q) = \{\{p\}, \{p, q\}\} \not\vdash -q \\ \text{while} & \{\{p\}\} \vdash -q \qquad \square \end{array}$$

Notice that by the addition of weak negation to (the query and input language of) Belnap's KS the inference operation is no longer persistent and the update operation no longer ampliative.

	$\boldsymbol{F}^+$	$\boldsymbol{F}$	$\boldsymbol{F}^-$	$\boldsymbol{B}$	$D_B\boldsymbol{F}$	$D_{ex}\boldsymbol{F}$	$D_{mi}\boldsymbol{F}$
Contraction	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Permutation	$\surd$	$\surd$		$\surd$	$\surd$	$\surd$	
Update Synchronicity	$\surd$	$\surd$		$\surd$	$\surd$	$\surd$	
Update Monotonicity	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	
input complete			$\surd$				
Lemma Redundancy	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Lemma Compatibility	$\surd$	$\surd$	$\surd$	$\surd$			
ampliative	$\surd$	$\surd$		$\surd$	$\surd$	$\surd$	$\surd$
persistent	$\surd$			$\surd$			
Unique Representation	$\surd$	$\surd$	$\surd$		$\surd$	$\surd$	$\surd$
non-explosive	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$		$\surd$
coherent	–			–		$\surd$	$\surd$
Disjunctive Syllogism	–	–	–			$\surd$	$\surd$

Table 3.1: Properties of some KSs.

Claim 6 (Three-Valued Logic) *Let $\models_3$ denote the consequence relation determined by proper partial (resp. three-valued) models wrt standard truth conditions (resp. strong Kleene tables), and $F, G \in L(\sim, \wedge, \vee)$. Then in $D_{ex}\boldsymbol{F}$, it holds that*

$$F \models_3 G \quad \textit{iff} \quad \mathsf{Upd}(0, F) \vdash G$$

Proof: If F does not have a proper partial model, then $\mathsf{Upd}(0, F) = \{\}$, and the assertion holds trivially. Otherwise every minimal proper partial model of F corresponds to a consistent element of $\mathrm{DNS}(F)$, hence $\mathrm{MinMod}_3(F) = \mathsf{Upd}(0, F)$, and the same argument as in claim 2 applies. □

3.4.1 Adding an S5-like Modal Operator for Definite Belief

The query language of $D\boldsymbol{F}$ can be extended by adding the modal operator B for definite belief, yielding the *modal query extension* $D^5\boldsymbol{F}$. For $X \in Y \subseteq 2^{\mathrm{Lit}}$, we define:

$$\begin{array}{rlll}
(\mathsf{B}) & X \vdash \mathsf{B}F & \text{if} & \text{for all } X' \in Y : X' \vdash F \\
(\sim\mathsf{B}) & X \vdash \sim\mathsf{B}F & \text{if} & \text{for some } X' \in Y : X' \vdash \sim F \\
(-\mathsf{B}) & X \vdash -\mathsf{B}F & \text{if} & \text{for some } X' \in Y : X' \vdash -F \\
(-\sim\mathsf{B}) & X \vdash -\sim\mathsf{B}F & \text{if} & \text{for all } X' \in Y : X' \vdash -\sim F
\end{array}$$

In this setting, two different ‘possibility’ operators are definable:

$$\begin{array}{rcl}
\mathrm{E}F & := & \sim\mathsf{B}\sim F \\
\mathrm{C}F & := & -\mathsf{B}\sim F
\end{array}$$

$\mathrm{E}F$ captures the situation where there is some evidence in favour of F, whereas $\mathrm{C}F$ captures the situation where $\sim F$ cannot be inferred, i.e. where F is – in this sense – consistent.

Formally,

$$D^5_B\boldsymbol{F}^- := \langle \{\emptyset\}, \leq, \mathrm{Fin}(2^{2^{\mathrm{Lit}}}), \vdash, L(-, \sim, \wedge, \vee, |, \mathsf{B}, \mathsf{E}, \mathsf{C}), \mathsf{Upd}_B, L^*(-, \sim, \wedge, \vee) \rangle$$

Observation 14 *The modal operators B and E are persistent, whereas C is not.*

Observation 15 (Representing and Querying the Unknown) *While the problem whether some fact p is unknown in a fact base can be simply decided by answering the query $-p \wedge - \sim p$ this does not work in a disjunctive fact base. Instead, the indeterminacy query can be reformulated in three different versions which are also discussed in [Thi92] where they are named 'simple', 'strong', and 'semi-strong ignorance', respectively. In the following three examples we express complete knowledge, $\mathsf{B}p$, about p, and three different forms of incomplete knowledge, $-\mathsf{B}q \wedge -\mathsf{B}\sim q$, about q:*

1. *Complete ignorance wrt q represented by*

$$\{\{p\}\} \vdash p \wedge -q \wedge - \sim q$$

2. *Strong ignorance wrt q:*

$$\{\{p\}, \{p, q\}, \{p, \sim q\}\} \vdash \mathsf{B}p \wedge \mathsf{E}q \wedge \mathsf{E} \sim q$$

3. *Semi-strong ignorance wrt q:*

$$\{\{p\}, \{p, \sim q\}\} \vdash \mathsf{B}p \wedge \mathsf{E} \sim q$$

or symmetrically,

$$\{\{p\}, \{p, q\}\} \vdash \mathsf{B}p \wedge \mathsf{E}q$$

3.5 Exact Predicates and the Closed-World Assumption

In a VKS, three kinds of predicates can be distinguished. The first distinction, proposed by Körner in [Kör66], reflects the fact that many predicates (especially in empirical domains) have truth value gaps: neither $p(c)$ nor $\sim p(c)$ has to be the case for specific instances of such *inexact* predicates, like, e.g., color attributes which can in some cases not be determined because of vagueness.

Other predicates, e.g. from legal or theoretical domains, are *exact*, and we then have, for instance, $m(S) \vee \sim m(S)$ and $even(1) \vee \sim even(1)$, stating that Susan is either married or unmarried and that 1 is either an even or an uneven number. Only exact predicates can be totally represented. Therefore, the predicates subject to the CWA in a KB form a subset of EXACT, the set of all predicates declared as exact in KB. For an exact predicate p and any term t from its domain, the resp. instance of the *tertium non datur (TND)* holds:

$$\mathrm{KB} \vdash p(t) \vee \sim p(t)$$

no matter whether p is subject to the CWA or not. For a non-CWA predicate p, this can only be achieved by adding the resp. TND instances to the KB (which is not possible in definite KSs where the input language does not allow for $\vee$).

Definition 7 (VKB) *A vivid KB, in its general form, is a quadruple*

$$\langle \mathit{EXACT}, \mathit{CWA}, \mathrm{KB}, \mathit{IC} \rangle$$

where EXACT is the set of exact predicates, $\mathit{CWA} \subseteq \mathit{EXACT}$ specifies the predicates subject to the Closed-World Assumption, and $\mathit{IC} \subseteq L_{\mathrm{Query}}$ is a set of integrity constraints which *have to be satisfied by any evolving state of the VKB.*[6] *A query F is inferable from a VKB if it can be derived from the TND completion of CWA(KB) wrt $EXACT$ and CWA. Formally,*

$$\begin{array}{l} \langle\, EXACT,\, CWA,\, \mathrm{KB}\,\rangle \vdash F \quad :\Longleftrightarrow \\ \qquad \mathsf{Upd}(\, CWA(\mathrm{KB}),\ \{p(t) \vee \sim p(t) : p \in EXACT - CWA,\, t \in dom(p)\}) \ \vdash\ F \end{array}$$

where $dom(p)$ denotes the domain of the predicate p.

Notice that in KBs of definite KSs, like fact bases, it is not possible to declare exact predicates not subject to the CWA. Therefore, in definite KSs, EXACT = CWA.

Claim 7 (Classical Logic) *If all predicates are assumed to be exact, the resp. TND completion of a disjunctive fact base leads half way to classical logic. Let $F, G \in L(\sim, \wedge, \vee)$, and $\models_2$ denote classical consequence. Then, if all predicates occuring in F and G are exact, $EXACT = Pred(F,G)$, and CWA is empty,*

$$F \models_2 G \quad \textit{iff} \quad \langle\, EXACT,\, \emptyset,\, \mathsf{Upd}(0,F)\,\rangle\ \vdash\ G$$

holds in $D_{ex}\boldsymbol{F}$, and it also holds in $D_{mi}\boldsymbol{F}$ provided that F is consistent.

3.5.1 Representing Three Kinds of Predicates

Among the exact predicates one can further distinguish between those which are totally represented in a KB and those which are not. Exact and totally represented predicates are subject to the CWA. For example, the local KB of some city knows all residents of the city, i.e. the CWA holds for *resident*, but it does not know of every resident whether (s)he is married or not because (s)he might have married in another city and this information is not present. Consequently, the CWA does not apply to *married* in this KB.

The CWA helps to reduce disjunctive complexity which is exponential in the number of exact non-CWA predicates: if n is the number of unknown ground atoms which can be formed by means of predicates declared as exact but not subject to the CWA in VKB, then VKB contains implicitly 2^n possible situation descriptions.

We illustrate these distinctions with an example. Let m, r, b, l denote the predicates *married*, *resident*, *blonde* and *is_looking_at*, and let M, P, S stand for the individuals *Mary*, *Peter* and *Susan*.

$$\mathrm{VKB} \quad = \quad \left\{ \begin{array}{l} \mathrm{EXACT} = \{m, r\} \\ \mathrm{CWA} = \{r\} \\ m(M),\, r(M),\, b(M),\, \sim m(S),\, \sim b(S),\, l(M,P),\, l(P,S) \end{array} \right.$$

The interesting queries we can ask this VKB and the resp. answers are:

1. Does a married person look at an unmarried one ? Yes, but the KB does not know who:

$$\begin{array}{l} \mathrm{VKB} \vdash \exists x, y : l(x,y) \wedge m(x) \wedge \sim m(y) \\ \mathsf{Ans}(\mathrm{VKB},\, l(x,y) \wedge m(x) \wedge \sim m(y)) = \{\{\langle M,P\rangle,\, \langle P,S\rangle\}\} \\ \mathrm{VKB} \not\vdash \exists x, y : \mathsf{B}(l(x,y) \wedge m(x) \wedge \sim m(y)) \end{array}$$

[6] We shall not discuss integrity constraints in this paper, and therefore drop the IC component.

2. Does a resident look at a non-resident ? Yes, Mary at Peter.

$$\mathsf{Ans}(\text{VKB}, l(x,y) \wedge r(x) \wedge {\sim} r(y)) = \{\{\langle M, P\rangle\}\}$$

3. Does a blonde person look at a non-blonde one ? No. The VKB is completely ignorant about the blondeness of Peter: neither is he blonde, nor is he non-blonde, nor is he blonde or non-blonde (he might be in-between):

$$\mathsf{Ans}(\text{VKB}, l(x,y) \wedge b(x) \wedge {\sim} b(y)) = \emptyset$$

Notice that the explicit TND completion of KB wrt EXACT would yield $2^3 = 8$ possible situation descriptions, which are reduced to $2^1 = 2$ by the CWA declaration of *resident*.[7]

3.6 The Disjunctive Extension of a KS

We now generalize the previous discussion of specific disjunctive knowledge systems. The *disjunctive extension* $D\boldsymbol{K}$ of a definite knowledge system

$$\boldsymbol{K} = \langle 0, \leq, L_{\text{KB}}, \vdash, L_{\text{Query}}, \mathsf{Upd}, L_{\text{Input}} \rangle$$

is defined as the KS

$$D_B\boldsymbol{K} = \langle \{0\}, \leq_d, \text{Fin}(2^{L_{\text{KB}}}), \vdash_d, L^d_{\text{Query}}, \mathsf{Upd}_B, L^d_{\text{Input}}\rangle$$

where the information ordering is given by

$$\begin{array}{lll} Y_1 \leq_d Y_2 & :\Longleftrightarrow & \forall X_2 \in Y_2 \exists X_1 \in Y_1 : X_1 \leq X_2, \text{ and} \\ & & \text{either (i)} \quad Y_1 \subseteq Y_2, \\ & & \text{or (ii)} \quad \exists X_1' \in Y_1 \forall X_2 \in Y_2 : X_1' \not\geq X_2 \end{array}$$

and for finite $Y \subseteq L_{\text{KB}}$,

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

In the sequel, we will ommit d as superscript, resp. subscript if it is clear from context. An element $X \in Y$ of a disjunctive KB Y is called a *possible situation description*.

Observation 16 *If $\langle L_{\text{KB}}, \leq\rangle$ is a preorder, then $\langle 2^{L_{\text{KB}}}, \leq_d\rangle$ is a partial order.*

Conjunction and disjunction can be added to the input language of $\boldsymbol{K}$, $L^d_{\text{Input}} := L^*_{\text{Input}}(\wedge, \vee)$, excluding again inputs with literals being in the scope of both an even and an odd number of weak negations,[8] and the update operation can be inductively extended in the following way:[9]

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

Besides the liberally paraconsistent system $D_B\boldsymbol{K}$ we also get $D_{mi}\boldsymbol{K}$ and $D_{ex}\boldsymbol{K}$ by using the update operations $\mathsf{Upd}_{mi}(Y, F) := \text{mi}(\mathsf{Upd}_B(Y, F))$, and $\mathsf{Upd}_{ex}(Y, F) := \text{Cons}(\mathsf{Upd}_B(Y, F))$.

---

[7] Without the CWA there are 3 tertium non datur disjunctions formed with $m(P)$, $r(S)$, and $r(P)$, while with the CWA there is only one such disjunction: $m(P) \vee {\sim} m(P)$.

[8] This guarantees that Permutation holds, and consequently that $(U\wedge)$ is well-defined.

[9] Recall that e stands for an extended literal (l or $-l$). All other inductive cases are handled by the above rewrite rules for complex formulas.

3.6.1 Efficient Representation

In 'real world' KBs the amount of disjunctive information can be expected to be small (and for reasons of computational efficiency has to be small) compared to the amount of definite information. Therefore, one should seek for possible optimizations, such as representing a disjunctive KB as a pair $\langle X, Y \rangle$ where $X \in L_{\mathrm{KB}}$ contains the definite and $Y \subseteq L_{\mathrm{KB}}$ the disjunctive information, and for any persistent query formula F,

$$\langle X, Y \rangle \vdash F \quad :\Longleftrightarrow \quad \begin{array}{ll} (i) & X \vdash F, \text{ or} \\ (ii) & X \cup X' \vdash F \text{ for all } X' \in Y \end{array}$$

Thus, if $Y \subseteq L_{\mathrm{KB}}$ is a disjunctive KB, then it can be 'folded' into

$$\langle \bigcap Y, \{X - \bigcap Y : X \in Y\} \rangle$$

3.7 Conclusion

We have presented natural extensions of Belnap's system of epistemic states. First, the addition of weak negation to the query language leads to unique representations and renders it possible to express exclusive disjunctive information. Second, the adoption of the principle of minimal inconsistency leads to Coherence and the Disjunctive Syllogism. Third, the addition of exactness declarations for specific predicates leads to a kind of local Reasoning by Cases.

Chapter 4

Towards an Operational Semantics of Rules

We develop a general closure semantics for deduction rules in knowledge bases, using the concepts of a *(rule) knowledge representation and reasoning system* proposed in [Wag94a] where rules are interpreted as update functions operating on knowledge bases. We define the notion of a *supported closure*, and we show that every ampliative rule 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 rule knowledge bases (including normal, extended, disjunctive, and other logic programming and rule-based systems). The stable closure semantics can be viewed as a generalization of the stable model semantics of [GL88], since the stable closures of a normal logic program coincide with its stable models.

This chapter is based on joint work together with M.A. Khamsi and D. Misane.

4.1 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 [Wag91b], 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,

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

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* **N**, and indeed, as shown in [Pea93], a rule of an extended logic program can be interpreted as a constructive implication in **N**. However, rules operating on disjunctive information, such as in

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

differ from constructive implication: in **N**, we would obtain r (the corresponding inference principle is called 'Disjunction in the Premise'), whereas a genuine rule-oriented semantics would not 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 representation and reasoning system*, and that of a *rule knowledge base (RKB)*, or *deductive knowledge base*, 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 RKBs 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 RKB. 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 RKB 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 RKBs 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 RKBs. While certain 'well-behaved' RKBs have a unique stable closure, there may be several stable closures, or none, in the general case.[3]

Stable closures generalize stable models [GL88], resp. answer sets [GL90], in essential respects: they are not only defined for logic programs with negation-as-failure, but for a far more general class of non-monotonic rule-based systems. Thus, the stable model semantics can be viewed as the proper specialization of the stable closure semantics for normal logic programs. It is not clear at the moment whether a similar generalization is possible for the competing well-founded semantics. If such a generalization is not possible for the well-founded semantics, this would be a strong argument in favour of the stable model semantics

[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 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 rule 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.

by indicating that it is the theoretically more general, and in this sense superior concept.

The structure of the paper is as follows: in section 2 we present some basic concepts of the theory of knowledge representation proposed in [Wag94a]. In section 3, the concept of a deductive knowledge base is introduced, and the stable closure semantics is defined. It is shown that the stable closures of a normal logic program are exactly its stable models. Also, a generalization of the immediate consequence operator is presented, and the proposed operational semantics of rules is compared with the more common 'relational' semantics. In sections 4–6, various cases of RKBs satisfying certain conditions are discussed. In section 7, we show that every ampliative RKB has a supported closure. We then present a variety of examples illustrating the problem of stratification. Finally, in section 9, we briefly discuss the question of non-ampliative rules by means of two examples.

4.2 Deductive Knowledge Bases

Rule knowledge bases, or *deductive knowledge bases*, as we prefer to call them here, were defined in [Wag94a]. Every KS $\boldsymbol{K} = \langle\, 0, \leq, L_{\mathrm{KB}}, \vdash, L_{\mathrm{Query}}, \mathsf{Upd}, L_{\mathrm{Input}}\,\rangle$ can be naturally extended to an *deduction-rule-based* system $R\boldsymbol{K}$. In $R\boldsymbol{K}$, a knowledge base $X \in L_{\mathrm{KB}}$ is supplemented by a set $R \subseteq L^x_{\mathrm{Input}} \times L^x_{\mathrm{Query}}$ of deduction rules $r = \langle F, G\rangle$ with conclusions $F \in L^x_{\mathrm{Input}}$ and premises $G \in L^x_{\mathrm{Query}}$, also written as '$F \leftarrow G$', such that $\mathsf{Free}(F) \subseteq \mathsf{Free}(G)$. Rules of this form are called *range-restricted*,[4] the set of such rules will be denoted by $R(L_{\mathrm{Input}} \leftarrow L_{\mathrm{Query}})$. They can be viewed as mappings between KBs, $r : L_{\mathrm{KB}} \to L_{\mathrm{KB}}$, since their application is defined as[5]

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

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. If $r = F \leftarrow G$ 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 17 *If the 'facts' X of a deductive knowledge base $\langle X, R\rangle$ consist of a set of compiled input formulas $A_X \subseteq L_{\mathrm{Input}}$, $X = \mathsf{Upd}(0, A_X)$, 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'.

---

[4] In fact, we also need to require that the premise of a rule can be sensibly evaluated. Query formulas with this property are called *domain independent*, resp. *evaluable*. See [vGT91]. In the special case of normal logic programs it means that every free variable occuring in the scope of negation-as-failure occurs as well in a positive literal.

[5] Notice that $r(X) = \mathsf{Upd}(X, \{F(t) : \{t\} \in \mathsf{Ans}(X, G)\})$. A generalized definition of rule application would be $r(X) = \mathsf{Upd}(X, \{\bigvee_{t \in T} F(t) : T \in \mathsf{Ans}(X, G)\})$. This is a topic of future research.

Definition 8 (Deductive Closure) *$Z \in L_{\mathrm{KB}}$ 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 $r \in R$.*[6]

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

Observation 18 *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,

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

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:

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

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 will 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 $R\boldsymbol{K}$ is the same as that of $\boldsymbol{K}$, and we have

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

Formally,

$$R\boldsymbol{K} \;:=\; \langle\, \langle 0, \emptyset\rangle,\, \leq_r,\, L_{\mathrm{KB}} \times 2^{R(L_{\mathrm{Input}} \leftarrow L_{\mathrm{Query}})},\, \vdash_r,\, L_{\mathrm{Query}},\, \mathsf{Upd}_r,\, L_{\mathrm{Input}}\,\rangle$$

Since we do not need it here, we will not discuss the definition of $\leq_r$.[7] In the sequel, we will omit the subscript r.

Example 4 (Normal Logic Programs) *Deductive knowledge bases of $R\boldsymbol{A}$ correspond to deductive databases, resp. normal logic programs. For instance, the program*

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

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

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

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

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

has the unique informationally minimal closure

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

[6] Notice that the notion of a closure is more general than a 'model'. In specific cases, such as for normal or extended logic programs, closures can be viewed as 'models'.

[7] Possibly: $\langle X, R\rangle \leq_r \langle X', R'\rangle \quad :\Longleftrightarrow \quad \forall F \in L_{\mathrm{Input}} : R(\mathsf{Upd}(X, F)) \leq R'(\mathsf{Upd}(X', F))$.

Example 5 (EDLPs) *An extended disjunctive logic program (EDLP) corresponds to a deductive knowledge base of* $RD\boldsymbol{F}$. *For instance,*

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

corresponds to the deductive knowledge base $\langle Y_2, R_2\rangle$, *where* Y_2 *is as in example 3, and* $R_2 = \{b(x) \leftarrow l(P,x)\}$, *expressing that every woman Peter likes is blonde, and thus,* $R_2(Y_2) = \{\{b(L), l(P,L), {\sim}b(S)\}\}$.

Definition 9 *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 19 *If Update Monotonicity holds in a KS, a rule is monotonic whenever its premise is persistent.*

Proof: Let $X_1 \leq X_2$. We have to distinguish two cases: If r is not applicable in X_1, then – since it is persistent – it is neither applicable in X_2, and hence

$$r(X_1) = X_1 \leq X_2 = r(X_2)$$

Otherwise, by Update Monotonicity,

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

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 10 (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* $(r_i)_{1\leq i<\alpha}$, *such that its composition,* $r = \bigcirc_{\alpha>i\geq 1} r_i$, *computes* Z: $Z = r(X)$.[8]

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, and 2) that all rules in R are ampliative (both conditions are satisfied, for instance, in $R\boldsymbol{A}$, and $R\boldsymbol{F}$). 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 $i \leq j < \beta$.
   (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, $X_{\beta-1} \leq X_\beta$.

[8] 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.

(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,

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

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 6 *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.

Definition 11 (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 with $\alpha > n \geq k$.

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

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

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.

[9] In fact, if there is the possbility of inconsistent KBs in K, then the preferred closures will be the *minimally inconsistent* stable closures according to some measure of inconsistency associated with K.

4.2.1 Stable Closures and Stable Models

Claim 8 *A stable model of a normal logic program is a stable closure of the corresponding $R\boldsymbol{A}$ knowledge base.*

Proof: Let M be a stable model of Π, i.e.

$$M = \Pi^M(\emptyset) = \bigcirc_{\alpha > i \geq 1} r_i(\emptyset)$$

for certain rules $r_i \in \Pi^M = \{a \leftarrow F^+ : a \leftarrow F \in \Pi \ \ \& \ \ F^- \cap M = \emptyset\}$. Each of these rules r_i is of the form $a \leftarrow F^+$, such that it is expandable to a rule $r'_i = a \leftarrow F \in \Pi$, and $F^- \cap M = \emptyset$. Therefore, M is also a supported closure of Π: $M = \bigcirc_{\alpha > i \geq 1} r'_i(\emptyset)$. It remains to show that this closure is stable. Abbreviating $M^i := r_i \circ \ldots \circ r_1(\emptyset)$, we have to show for arbitrary $k \in [1, \alpha)$ that $M^n \vdash P_{r_k}$ for all $n \in [k, \alpha)$. Since $P^+_{r_k} \subseteq M^{k-1} \subseteq M^n$, it holds that $M^n \vdash P^+_{r_k}$. On the other hand, since $M \cap P^-_{r_k} = \emptyset$, it also holds that $M^n \cap P^-_{r_k} = \emptyset$. Consequently, $M^n \vdash P_{r_k}$. □

Claim 9 *A stable closure of a $R\boldsymbol{A}$ knowledge base $\langle X, R\rangle$ is a stable model of the corresponding normal logic program Π.*

Proof: Let $Z = \bigcirc_{\alpha > i \geq 1} r_i(X)$ be a stable closure of $\langle X, R\rangle$. For every $k \in [1, \alpha)$, $r'_k = C_{r_k} \leftarrow P^+_{r_k} \in \Pi^Z$, since $Z \vdash P_k$ implies that $P^-_k \cap Z = \emptyset$. Consequently, $Z \subseteq \Pi^Z(\emptyset)$.

Now, assume that $\Pi^Z(\emptyset) \supset Z$, i.e. that there are further applicable rules $\{r_\alpha, r_{\alpha+1}, \ldots\} \subseteq \Pi^Z$, such that $\Pi^Z(\emptyset) = \bigcirc_{\beta > i \geq \alpha} r_i(Z)$. Then $Z \vdash P_{r_\alpha}$, but $C_{r_\alpha} \notin Z$, i.e. Z is not closed under r_α which contradicts the fact that Z is a closure of Π. Consequently, $\Pi^Z(\emptyset) \subseteq Z$. □

4.2.2 The Immediate Consequence Operator

Definition 13 *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 $R\boldsymbol{A}^+$, one obtains the classical T_P-operator : let $\langle X, R\rangle$ be the RKB 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 20 *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 21 *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 22 *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 10 *If Update Synchronicity holds, then every stable closure 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 22, Z is a fixpoint of $T_{X,R}$. By observation 21, Z is also a fixpoint of T_R. □

4.2.3 Operational and Relational Semantics of Rules

Traditionaly, rules have been interpreted by means of model-theory in a similar way as conditionals, or even as material implication. Starting with some notion of a model-theoretic interpretation of formulas, one says that a rule is satisfied by an interpretation if its conclusion is satisfied whenever its premise is. An interpretation is called a model of a set of rules R if it satisfies all $r \in R$. As usual, one can then define a formula to be a consequence of R, if it is satisfied by all (preferred) models of R. In the same spirit can we now define a *relational closure* of a deductive knowledge base.

Definition 14 (Relational Closure) *A relational closure of $\langle X, R\rangle$ is a KB $Z \geq X$, such that for all $r \in R$, $Z \vdash C_r$ whenever $Z \vdash P_r$. We denote the set of all relational closures of $\langle X, R\rangle$ by $R^{\vdash}(X)$.*

One might ask now: under which conditions does the relational semantics of rules aggree with the operational one proposed above ?

Observation 23 *A relational closure is an operational (deductive) closure if Vacuity holds.*

Proof: Let Z be a relational closure of $\langle X, R\rangle$, and let $r \in R$. It suffices to show that $r(Z) = Z$. If $Z \not\vdash P_r$, then, by definition, $r(Z) = Z$. On the other hand, if $Z \vdash P_r$, then $Z \vdash C_r$, and by Vacuity, $r(Z) = \mathsf{Upd}(Z, C_r) = Z$. $\Box$

Observation 24 *An operational closure is a relational closure if Reflexivity, i.e.* $\mathsf{Upd}(\mathrm{KB}, F) \vdash F$ *for all* $F \in L_{\mathrm{Input}}$*, holds.*

Proof: Let Z be an operational closure of $\langle X, R\rangle$, and let $r \in R$ such that $Z \vdash P_r$. Then $r(Z) = \mathsf{Upd}(Z, C_r) = Z$, and by Reflexivity, $Z \vdash C_r$. $\Box$

Reflexivity does not hold in general in knowledge systems because certain inconsistency handling mechanisms may exclude inconsistent inputs from reflexivity. Vacuity and Reflexivity hold, e.g., in $\boldsymbol{A}$, and therefore our operational semantics of rules aggrees with the relational semantics of [GL88] in the case of normal logic programs. For more complicated KSs, however, such as $D\boldsymbol{F}$, where Vacuity does not hold, no such aggreement can be expected.

Our concept of an operational closure seems to have some advantages when compared with the concept of a relational closure, resp. the traditional model-theoretic semantics of rules:

- In the operational semantics of rules we have a general straightforward notion of supported closure. In the relational semantics, the concept of a supported closure – in the form of supported models as proposed in [ABW88] – is only possible for simple structures (like classical, or partial models); it does, however, not guarantee 'groundedness', i.e. an atom can support itself.

- The operational semantics allows for the notion of a stable closure as a preference criterion (without having to refer to any specific operator, such as negation-as-failure) while the relational semantics can only distinguish the minimal (relational) closures as the preferred ones which is not sufficient.

On the other hand, the relational semantics seems to be, in some sense, more general: it allows for query expressions in the conclusion of a rule, while the operational semantics is restricted to input expressions.

Example 7 *There are cases where only a relational, but no operational, closure exists; e.g. the relational closure of the* $RD\boldsymbol{F}1$ *knowledge base* $\langle\{\{p\}\}, \{p|q \leftarrow 1\}\rangle$ *is* $\{\{p\}\}$*, while it has no operational closure.*

4.3 Monotonic RKBs

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 $r \in R$ have persistent premise formulas (see observation 19).[10]

[10] Notice that we do not require in this section that the KS is monotonic, i.e. persistent and ampliative, but only that the RKB is monotonic.

Claim 11 *A monotonic RKB 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, $\hat{X} \leq Z$. □

4.4 Persistent Ampliative RKBs

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,

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

Claim 12 *For a monotonic KS with Update Synchronicity, the operator $T_{X,R}$ is monotone.*

Proof: Let $Z_1 \leq Z_2$. Because of Persistence 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 Upd is ampliative, we get $T_{X,R}(Z_1) \leq T_{X,R}(Z_2)$. □

Claim 13 *Any supported closure of a persistent ampliative RKB 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 RKB $\langle X, R \rangle$ the operator $T_{X,R}$ is monotone, the
1 corresponding fixpoint set $Fix(T_{X,R})$ is a complete sublattice. Hence there exists a least
2 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 14 *Let $\langle X, R \rangle$ be a persistent ampliative RKB. 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 15 *In a KS satisfying Update Synchronicity and where the information ordering is a partial order, any supported closure of a persistent ampliative RKB is equal to $\hat{X}$. Therefore every persistent ampliative RKB has a unique supported closure.*

Proof: Let Z be a supported closure of a persistent ampliative RKB $\langle X, R\rangle$. 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.

4.5 Monotonic Ampliative RKBs

Observation 25 *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,

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

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 16 *A monotonic ampliative RKB has a unique supported closure which is the least closure.*

Proof: Let $\langle X, R\rangle$ be a monotonic ampliative RKB. 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 $\hat{X} \leq Z$. $\square$

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 RKBs and the other one for monotonic ampliative RKBs. They can be combined in the case of persistent ampliative RKBs of a KS satisfying Update Monotonicity.

4.6 Ampliative RKBs

Typical examples for non-persistent ampliative RKBs are normal logic programs with negation-as-failure. Positive logic programs (without negation-as-failure) are monotonic, persistent and ampliative RKBs. Extended disjunctive logic programs without weak negation (corresponding to RKBs of $RD\boldsymbol{F}$) are persistent and ampliative, but not monotonic RKBs.

Claim 17 *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 RKB. 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 RKBs is similar to that of normal logic programs. Although they always have supported, resp. minimal models, we are rather interested in their stable models. Therefore, an important question is: under which conditions does an ampliative RKB 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 the second part of our paper.

We call a RKB *unstable* if it has no stable model, otherwise it is called *stable*.

4.6.1 Some Examples of Stable RKBs

Example 8 *The program $\Pi_2 = \{p \leftarrow -q,\ q \leftarrow -p\}$ has two stable closures, $\{p\}$ and $\{q\}$, and thus is indefinite, i.e. it represents the disjunctive information $p|q$. This aggrees with the stable semantics.*

Example 9 *The $RD\boldsymbol{F}$ knowledge base $\langle\{\{\sim p\}, \{s\}\}, \{p \leftarrow -q, q \leftarrow -p\}\rangle$ has two stable closures, $\{\{p, s\}\}$ and $\{\{\sim p, q\},\ \{q, s\}\}$, which represent exactly the three answer sets of the corresponding EDLP $\{\sim p \vee s,\ p \leftarrow -q,\ q \leftarrow -p\}$.*

Example 10 *The normal program $\Pi_3 = \{p \leftarrow -p,\ p \leftarrow -q,\ q \leftarrow -p\}$ has two supported closures, $\{p\}$ and $\{p, q\}$. Its unique stable closure is $\{p\}$.*

Example 11 *The normal program $\Pi_6 = \{p \leftarrow -q,\ p \leftarrow q,\ r \leftarrow -q,\ q \leftarrow -r\}$ has two supported closures: $\{p, q\}$, and $\{p, r\}$, which are also its stable closures and its stable models.*

4.6.2 Unstable RKBs

We only discuss some typical examples of unstable RKBs. Normal (resp. consistent extended) logic programs without stable models (resp. answer sets) are also called 'incoherent' in the literature.

Example 12 *The $R\boldsymbol{A}$ knowledge base $\langle\emptyset, \{p \leftarrow -p\}\rangle$, corresponding to the well-known* call-inconsistent*logic program $\Pi_1 = \{p \leftarrow -p\}$, has the unique supported closure $\{p\}$, but no stable closure. In the logic programming literature Π_1 is considered to be either empty (according to the well-founded semantics) or even 'incoherent' (according to the stable semantics). The unique minimal model of Π_1 is $\{p\}$.*

Example 13 *The $RD\boldsymbol{F}$ knowledge base $\langle\{\emptyset\}, \{\sim p \vee q \leftarrow 1,\ p \leftarrow -p\}\rangle$ has the unique supported closure $\{\{p, q\}\}$, while it does not have a stable closure, and the corresponding EDLP does not have an answer set.*

Example 14 *The program $\Pi_4 = \{p \leftarrow -q,\ q \leftarrow p\}$ has exactly one supported closure: $\{p, q\}$. It has neither a stable closure nor a stable model.*

Example 15 *The unique supported closure of $\Pi_5 = \{p \leftarrow q,\ p \leftarrow -q,\ q \leftarrow p,\ q \leftarrow -p\}$ is $\{p, q\}$. Π_5 has neither a stable closure nor a stable model.*

4.7 Non-Ampliative RKBs

Little is known on non-ampliative RKBs. And it does not seem to be clear whether non-ampliative inference rules can be useful at all in knowledge representation. We only discuss some examples.

Example 16 (Reductive Rules) *Rules in $R\boldsymbol{A}^-$ 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 RKBs 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 inconsistent. Reductive rules, therefore, do not seem to make sense in rule knowledge bases: either they are redundant, or inconsistent.

Example 17 (Non-Ampliative Non-Reductive Rules) *In $D\boldsymbol{F}^-$, a set of extended literals, $E \subseteq \mathrm{XLit}$, as input is processed according to*

$$(UE) \qquad \mathsf{Upd}(Y, E) = \{X \cup (E^+ - E^-) - (E^- - E^+) : X \in Y\}$$

A disjunction $\bigvee_{i=1}^n E_i$ of concjunctions of extended literals E_i is processed according to

$$(U\bigvee) \qquad \mathsf{Upd}(Y, \bigvee_{i=1}^n E_i) = \bigcup_{E \in \mathcal{E}^*} \mathsf{Upd}(Y, E)$$

where $\mathcal{E}^* = mi(2^{E_1 \cup \ldots \cup E_n})$, and for $\mathcal{E} \subseteq 2^{\mathrm{XLit}}$,

$$mi(\mathcal{E}) = \{E \in \mathcal{E} : (F \cap \overline{F}) \not\subset (E \cap \overline{E}) \text{ for all } F \in \mathcal{E}\}$$

Exclusive disjunction can be defined in $D\boldsymbol{F}^-$ *as* $F|G := (F \wedge -G) \vee (G \wedge -F)$. *We could then have the following non-ampliative rule set* $R = \{q|r \leftarrow -p\}$, *which is non-redundant in cases like* $Y = \{\emptyset\}$,

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

and inconsistent in certain other cases like $Y = \{\{q, r\}\}$, *which cannot be closed under* R.

4.8 Conclusion

We have presented a general semantical framework for (possibly nonmonotonic) deduction rules in knowledge bases. 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. In the next chapter, it will be shown how the stable closure semantics can be adapted to active knowledge bases, thus establishing a link between deduction and action rules.

Chapter 5

The Stable Closure Semantics for Active Knowledge Bases

Abstract Active knowledge bases result from the addition of action rules to ordinary knowledge bases, like relational or deductive databases and their (for instance, disjunctive) extensions. Action rules provide, e.g., for knowledge base updates on certain conditions, thus beeing able to enforce integrity constraints. But they can also call procedures, like running certain diagnostic tests, or turning switches on and off. In this paper, we will focus on update actions. When updating an active KB, all action rules have to be checked and fired if their condition is satisfied, leading to subsequent updates triggering further action rules. This recursive process is captured declaratively by the notion of an active closure of a KB, being an appropriately updated KB closed under all action rules. Since there might be several closures, we define the notion of a ***stable closure*** of an active knowledge base in order to distinguish the preferred, resp. intended, ones from the non-intended ones. We are then able to show that an effectively stratified deductive database under the stable model semantics can be faithfully transformed into an active database under the stable closure semantics.

5.1 Introduction

The idea of active database management systems has arisen from different practical considerations, such as meeting the requirements of non-standard applications, or providing an effective means for the automatic enforcement of integrity constraints.[1] However, although the concept of action rules ('triggers') has gained considerable attention and widespread acceptance both in the academic and the industrial community, there seems to be a lack of formal foundation for it. In this paper, we present formal definitions of action rules and active knowledge bases which are more general than other approaches since they allow the addition of action rules to all kinds of ordinary knowledge bases, like relational or deductive databases and their (for instance, disjunctive) extensions. We also define a general semantics for active knowledge bases in terms of stable closures, and show that an effectively stratified deductive database under the stable model semantics of [GL88] can be faithfully transformed into an active database under the stable closure semantics. Our formalization and semantics relates action with deduction rules, and thus facilitates the further investigation of their resp. potential as components of advanced information systems.

[1] See, e.g., [Cha91].

In [Wag94a], the formal concept of a *knowledge representation and reasoning system (KS)* was introduced. A KS consists of two main components: an answer and an update operation manipulating knowledge bases as abstract objects. Based on this formal framework it was shown how with every KS $\boldsymbol{K}$ an active KS $A\boldsymbol{K}$ can be associated. The semantics of $A\boldsymbol{K}$ differs from $\boldsymbol{K}$ with respect to the update operation. When updating an active knowledge base, the updated KB has to be closed again under all action rules.

In some approaches to the formalization of updates and action rules, updates are explicitly recorded by means of two meta-predicates *inserted* and *deleted*, sometimes even in combination with explicit time stamps providing a full history of the knowledge base evolution (e.g., [Bry90], [LPS93], [Zan93]). Since this not only may obscure the semantics of a KB but also clearly creates a vast overhead of historic and meta-data, a more conservative approach where the result of an update is again a plain KB without any meta-data, as in our model of a KS, seems to be preferrable.

The concept of a *production rule system*, which has been widely used in AI for building 'expert systems', seems to be a special case of an active KS. In the literature on production rule systems (e.g. [Ras94]) and active databases (e.g. [Zan93]), update actions in the conclusion of a rule have been expressed by means of special operators like *add/delete*, or *assert/retract*, and the like. We propose a conceptually simpler, and more elegant, alternative. Instead of

$$\mathrm{retract}(p) \leftarrow q \wedge -r$$

we use the negation-as-failure operator '$-$' also in the head of a rule in order to express deletion, resp. contraction:

$$-p \leftarrow q \wedge -r$$

In this way, we obtain purely logical (and uniform) expressions both in the premise and the conclusion of a rule.

The paper has the following structure: in section 2, we present the basic concepts of the theory of knowledge representation proposed in [Wag94a]; in section 3, we present a formal semantics for action rules; in section 4, we show how deductive databases can be faithfully transformed into active databases, and in section 5, we illustrate the use of action rules for automatic integrity maintenance.

5.2 Active Knowledge Bases

Active knowledge bases (AKBs) were originally defined in [Wag94a] as a generalization of active databases. Here we improve and extend that presentation, and we define the stable closure semantics for AKBs.

AKBs result from the addition of *action rules* to ordinary KBs. Action rules have the form

$$\mathit{Action} \leftarrow \mathit{Condition}$$

In this paper we shall restrict our attention to rules with update actions and purely epistemic conditions. In general, however, actions will not be restricted to updates but also include other private actions, like turning certain switches on and off, and communicative and further cooperative actions, like informing other KBs, or requesting certain actions from other active KBs. Moreover, general conditions will not only refer to the current knowledge state (i.e.

X) but also to the occurence of certain events (such rules are called *event-condition-action rules*).

An active KB can be modeled as a pair $\langle X, R\rangle$ where $X \in L_{\text{KB}}$ is a knowledge base from some KS $\boldsymbol{K} = \langle 0, \leq, L_{\text{KB}}, \vdash, L_{\text{Query}}, \mathsf{Upd}, L_{\text{Input}}\rangle$, and $R \subseteq L^x_{\text{Input}} \times L^x_{\text{Query}}$ is a set of action 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 $\mathsf{Free}(F) \subseteq \mathsf{Free}(G)$, and G can be sensibly evaluated.[2] Rules of this form are called *range-restricted*, the set of such rules will be denoted by $R(L_{\text{Input}} \leftarrow L_{\text{Query}})$. They can be viewed 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. If $r = F \leftarrow G$ we 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 $[R]$.

An active knowledge base $\langle X, R\rangle$ is closed under all action rules: $X = r(X)$ for all $r \in R$. When updating an AKB, all action rules have to be checked and fired if their condition is satisfied, leading to subsequent updates. This recursive process is captured declaratively by the notion of an *active closure* of a KB. The action-rule-based extension of $\boldsymbol{K}$ is defined in the following way:

$$A\boldsymbol{K} \;:=\; \langle\, \langle 0, \emptyset\rangle, \leq_a, L_{\text{AKB}}, \vdash_a, L_{\text{Query}}, \mathsf{Upd}_a, L_{\text{Input}}\,\rangle$$

where

$$L_{\text{AKB}} = \{\langle X, R\rangle \in L_{\text{KB}} \times 2^{R(L_{\text{Input}} \leftarrow L_{\text{Query}})} : X = R(X)\}$$

The active update operation Upd_a is defined by means of the *active closure* $R(\mathsf{Upd}(X, F))$ of an updated knowledge base $\mathsf{Upd}(X, F)$:

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

In an active KB, an update event may cause additional (secondary) updates by triggering action rules. Therefore Upd_a has to close again the updated KB.

For the sake of simplicity, we assume that every $X \in L_{\text{KB}}$ has a unique preferred active closure, denoted by $R(X)$, according to the following definitions.[3]

Definition 15 (Active Closure) *$Z \in L_{\text{KB}}$ is called an* active closure *of $X \in L_{\text{KB}}$ under a set of action rules R, if*

1. *Z is closed under all rules $r \in R$, i.e. $r(Z) = Z$ for all $r \in R$, and*
2. *Z is supported, i.e. for some sequence of rules $(r_i)_{1 \leq i \leq n} \subseteq R$, $Z = r_n \circ \ldots \circ r_1(X)$.*

[2] Query formulas with this property are called *domain independent*, resp. *evaluable*. See [vGT91]. In the special case of a conjunction of negation-as-failure literals, it means that every free variable occuring in the scope of negation-as-failure occurs as well in a positive literal.

[3] The general case of possibly several preferred active closures will be treated in a future paper.

In the sequel, we shall also simply say 'closure' instead of 'active closure'.

Example 18 *Let $X = \emptyset$, and $R = \{-p \leftarrow -r, r \leftarrow q\}$ form an AKB of $A\boldsymbol{A}^-$, and $p \wedge q$ be a new input.* $\mathsf{Upd}(X, p \wedge q) = \{p, q\}$ *has two closures under R:*

$$\{q, r\} = (r \leftarrow q)(-p \leftarrow -r)(\{p, q\}) \tag{5.1}$$
$$\{p, q, r\} = (r \leftarrow q)(\{p, q\}) \tag{5.2}$$

However, only the latter is preferred, since it is the only stable closure *in the sense that each rule application within the computation sequence preserves the applicability of all previously applied rules.*

Definition 16 (Stable Closure) *A closure Z of a knowledge base $X \in L_{\mathrm{KB}}$ under a set of action rules R is called* stable *if there is a sequence of rules $(r_i)_{1 \le i \le n} \subseteq R$, such that its composition, $r = r_n \circ \ldots \circ r_1$, forms a stable computation of Z in the following sense: $Z = r(X)$, and*

$$\textit{If } r_{i-1} \circ \ldots \circ r_1(X) \vdash P_{r_i}, \textit{ then } r_j \circ \ldots \circ r_1(X) \vdash P_{r_i}, \textit{ for all } j \textit{ with } i \le j \le n.$$

If $Z = r_n \circ \ldots \circ r_1(X)$ is a stable closure of X under R, then we may assume (without loss of generality) that for every $i \le n$, and $j \le i$,

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

implying that $Z \vdash P_{r_j}$ for all $j = 1, \ldots, n$.

In this paper, we shall restrict our considerations to the case where a unique stable closure exists, and $R(X)$ is identified with its single element, i.e. we write $R(X) = Z$ instead of $R(X) = \{Z\}$. In general, however, there may be several (or no) stable closures, as the following example shows.[4]

Example 19 *Let $X = \emptyset$, and $R = \{-p \leftarrow q, -q \leftarrow p\}$ form an AKB of $A\boldsymbol{A}^-$, and $p \wedge q$ be a new input.* $\mathsf{Upd}(X, p \wedge q) = \{p, q\}$ *has two stable closures under R: $\{q\}$ and $\{p\}$.*

Definition 17 (Preferred Closures) *We define the preferred (intended) closures of an active knowledge base to be the stable ones:*[5]

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

Since the epistemic component X of an AKB $\langle X, R\rangle$, is always 'saturated', $X = R(X)$, the answering of a query does not involve the active component R:

$$\langle X, R\rangle \vdash_a F :\Longleftrightarrow X \vdash F$$

Notice that while the deductive closure of a rule knowledge base such as a deductive database, is an auxiliary object of the semantics (and only temporarily, or partially, computed) in order to determine the answer to a query, the active closure of an active knowledge base is effectively carried out as an essential part of the update operation of an active KS. In short: deduction rules help to answer queries, whereas action rules help to perform updates.

[4]Notice that there are four kinds of 'satisfiability': an AKB can have 1) a unique stable closure, 2) several stable closures, 3) a closure but no stable one, and 4) no closure at all, like for instance $\langle \emptyset, \{p \leftarrow -p, -p \leftarrow p\}\rangle$.

[5]In fact, if there is the possbility of inconsistent KBs in K, then the preferred closures will be the *minimally inconsistent* stable closures according to some measure of inconsistency associated with K.

5.3 Transforming Deductive into Active Databases

A deductive database Π consists of a set of ground atoms $X_\Pi \subseteq \mathrm{At}$, and a set R_Π of range-restricted deduction rules of the form $a \leftarrow A$ where $a \in \mathrm{At}^x$ is an atom, and $A \subseteq \mathrm{At}^x \cup \overline{\mathrm{At}^x}$ a set, resp. conjunction, of atoms and weakly negated atoms, such that $\mathsf{Free}(a) \subseteq \mathsf{Free}(A)$, and $\mathsf{Free}(A^-) \subseteq \mathsf{Free}(A^+)$.

$M \subseteq \mathrm{At}$ is a *stable model* of Π, if $M = \alpha(\Pi^M)$, where $\Pi^M = X_\Pi \cup R_\Pi^M$, and

$$R_\Pi^M = \{a \leftarrow A^+ : a \leftarrow A \in R_\Pi \;\;\&\;\; A^- \cap M = \emptyset\}$$

and α assigns the least model to a positive program.

Claim 18 *A stable model of a deductive database* Π *is a stable closure of* X_Π *under* R_Π.

Proof: Let M be a stable model of Π, i.e. $M = \alpha(\Pi^M) = r_n \circ \ldots \circ r_1(X_\Pi)$, for certain rules $r_i \in R_\Pi^M = \{a \leftarrow A^+ : a \leftarrow A \in R_\Pi \;\;\&\;\; A^- \cap M = \emptyset\}$. Each of these rules is expandable to a rule $r_i' = a \leftarrow A \in R_\Pi$, and since $A^- \cap M = \emptyset$, M is an active closure of X_Π under R_Π: $M = r_n' \circ \ldots \circ r_1'(X_\Pi)$. It remains to show that this closure is stable. Abbreviating $M^i := r_i \circ \ldots \circ r_1(X_\Pi)$, we have to show for arbitrary $j = 1, \ldots, n$ that $M^i \vdash P_{r_j}$ for all $i = j, \ldots, n$. Since $P_{r_j}^+ \subseteq M^{j-1} \subseteq M^i$, it holds that $M^i \vdash P_{r_j}^+$. On the other hand, since $M \cap P_{r_j}^- = \emptyset$, it also holds that $M^i \cap P_{r_j}^- = \emptyset$. Consequently, $M^i \vdash P_{r_j}$. □

Claim 19 *A stable closure of* $X \subseteq \mathrm{At}$ *under a set* R *of action rules of the form* $a \leftarrow A$ *is a stable model of the corresponding deductive database* $\Pi = X \cup R$.

Proof: Let $Z = r_n \circ \ldots \circ r_1(X) = X \cup \{C_{r_i} : 1 \leq i \leq n\}$ be a stable closure of X under R. For every $i = 1, \ldots, n$, $r_i' = C_{r_i} \leftarrow P_{r_i}^+ \in R_\Pi^Z$, since $Z \vdash P_i$ implies that $P_{r_i}^- \cap Z = \emptyset$. Consequently, $Z \subseteq \alpha(\Pi^Z)$.

Now, assume that $\alpha(\Pi^Z) \supset Z$, i.e. that there are further applicable rules $\{r_{n+1}, \ldots, r_m\} \subseteq R_\Pi^Z$, such that $\alpha(\Pi^Z) = r_m \circ \ldots \circ r_{n+1}(Z)$. Then $Z \vdash P_{r_{n+1}}$, but $C_{r_{n+1}} \notin Z$, i.e. Z is not closed under r_{n+1} which contradicts the fact that Z is a closure of X under R. Consequently, $\alpha(\Pi^Z) \subseteq Z$. □

Claim 20 *An effectively stratified*[6] *deductive database* Π *can be faithfully transformed into an active database* $A_\Pi = \langle R_\Pi(X_\Pi), R_\Pi\rangle$*, such that* Π *and* A_Π *are dynamically equivalent, i.e. for all* $p, q \in \mathrm{At}$,

$$\Pi \cup \{p\} \vdash q \quad \textit{iff} \quad \mathsf{Upd}(A_\Pi, p) \vdash q$$

Proof: An effectively stratified deductive database has a unique stable model. Since $\Pi \cup \{p\}$ is effectively stratified whenever Π is effectively stratified, it suffices to show that every stable model of Π is a stable closure of X_Π under R_Π, and vice versa, as has been done in the previous two claims. □

[6] *Effective stratification* has been defined in [BF91]. For an effectively stratified normal logic program, the wellfounded 'model' is equal to the unique stable model.

5.4 Some Remarks on Update and Revision

Traditional logics, like classical or intuitionistic first order logic, do not account for the *dynamics* of information processing which is most obvious in the case of the update operation associated with database and knowledge-based systems. Therefore, the semantics of updates has attracted considerable attention in recent years, both in the field of logics and AI (e.g., [Win 86+90], [Gär88], [KM91]), and in database and logic programming theory (e.g., [FUV83], [Abi88], [Rei92], [LPS93], [LPR93]). In logics and AI the primary concern seems to be quite theoretical, namely a formalization (or even axiomatization) of contraction and revision in terms of classical logic, resp. two-valued model theory. In contrast, in database and logic programming theory the primary concern is the operational modelling of insertion and deletion subsumed under the concept of update (as opposed to the more theoretical concept of revision which is based on a notion of inconsistency involving explicit negative information).

Observation 26 (AGM Update Postulates) *The following postulates for updates in a KS are adapted from analogues in [Gär88] where they have been formulated separately for knowledge base* expansion *and* contraction.

$$\begin{array}{rl} (\mathit{Success}) & \mathsf{Upd}(\mathrm{KB}, F) \vdash F \\ (\mathit{Vacuity}) & \mathrm{KB} \vdash F \;\Rightarrow\; \mathsf{Upd}(\mathrm{KB}, F) = \mathrm{KB} \\ (\mathit{Extensionality}) & F \vdash G \;\&\; G \vdash F \;\Rightarrow\; \mathsf{Upd}(\mathrm{KB}, F) = \mathsf{Upd}(\mathrm{KB}, G) \end{array}$$

The postulate of Success corresponds to the reflexivity *condition from Tarskis axioms for consequence relations, and also to KS2. Extensionality, in [Gär88], is a kind of soundness condition with respect to classical logic, whereas here we propose that every KS defines its own logic (whether classical or not).*

In a positive KS (not allowing for explicit negative information), Success holds for both insertion, e.g. $\mathsf{Upd}(\{p\}, q) \vdash q$, and for deletion, e.g. $\mathsf{Upd}(\{p, q\}, -q) \vdash -q$. This shows the conceptual advantage of treating weak negation (alias negation-as- failure) in an input formula as deletion, resp. contraction. In general, there may be KSs where Success does not hold for inconsistent inputs. Also, since not all KSs are cumulative, Vacuity does not hold in general. Extensionality, however, holds in all 'reasonable' systems.

5.5 Event-Specific Action Rules

In real life, many actions are caused by events. Also for action rules in a knowledge base it appears essential that they may be triggered by the occurence of certain events. Actions and events are closely related to each other: if some agent (say an AKB) performs an action (e.g. an update action), this creates an event (an update event). Since updates are basic actions, they are also basic events.[7]

We assume that an active KB can access information about those events it has to deal with by querying an event buffer. This event buffer is accessible by means of two system predicates PEvt and NEvt both taking an expression from an event language $L_{\mathrm{Evt}} \supseteq L_{\mathrm{Input}}$ as an argument. When a new event occurs, say an update, its description in the form of an event expression *Event* is placed in the event buffer where it must wait to be processed.

[7] For the sake of simplicity, we shall not distinguish between an action attempt and the (successfull) action itself.

Immediately before, resp. after, it is processed it can be accessed as by means of $\mathsf{NEvt}(\mathit{Event})$, resp. $\mathsf{PEvt}(\mathit{Event})$. At any system time point, PEvt is satisfied by the previous event, and NEvt is satisfied by the next event in the event buffer. After all action rules have been satisfied, *Event* is processed and the buffer pointer moves on to the next entry.

An event expression is either an elementary event, a compound event, a concurrent event (i.e. a set of compound events), or a sequence of events. For instance, in $AD\boldsymbol{A}^{-}$, elementary event expressions are extended literals corresponding to insertions and deletions, and compound events are compound input formulas. *Extended action rules* then have the form:

$$\mathit{Input} \leftarrow \mathsf{PEvt}(\mathit{PreviousEvent}), \mathsf{NEvt}(\mathit{NextEvent}), \mathit{Query}$$

where the premise is a conjunctive combination of an event query and an epistemic query from L_{Query} (each of which can be empty), such that update actions can depend both on the knowledge state of a KB and the information about currently occuring events.

The formal semantics for extended actions rules, i.e. the appropriate notion of a closure of a set of such rules, is a topic of future research.

5.6 Integrity Maintenance by Means of Action Rules

We shall discuss now the application of action rules for automatic integrity maintenance. In general, there are various kinds of integrity constraints defining the admissible KBs, and thus controlling knowledge base evolution, in a certain knowledge domain. Formally, an integrity constraint is a closed query formula $G \in L_{\text{Query}}$, such that an input F, and the corresponding update, is legal only if $\mathsf{Upd}(\text{KB}, F) \vdash G$.

In certain cases of updates $\mathsf{Upd}(\text{KB}, F)$, if an integrity constraint G is violated, one does not want to reject the update but rather to provide for an appropriate change of KB such that the cause of the violation is eliminated. This can be done by means of action rules, and one major application seems to be *referential integrity* maintenance.

5.6.1 Enforcing Integrity

We illustrate the use of action rules for automatic integrity maintenance by means of an example. Assume we have the information that Peter (P) is enrolled (enr) in math (m) or in philosophy (ph), and that it is not possible to do a master of science (msc) in philosophy. This is expressed in the following AKB:

$$\begin{aligned}\langle Y, R\rangle \;\; := \;\; & \langle\, \{\{enr(P,m)\},\, \{enr(P,ph)\},\, \{enr(P,m), enr(P,ph)\}\},\\ & \{-enr(x,ph) \leftarrow msc(x)\}\,\rangle\end{aligned}$$

If we then learn that Peter is doing a master of science, and update the AKB correspondingly, we have to compute

$$R(\mathsf{Upd}(Y, msc(P))) = \{\{enr(P,m)\}\}$$

and consequently, $\mathsf{Upd}(\langle Y, R\rangle,\, msc(P)) \;\vdash\; enr(P,m)$.

5.6.2 Referential Integrity in RDBs

SQL databases allow for the specification of a particular form of action rule enforcing *referential integrity* by automatically deleting all dependent tuples when a master tuple, to which these dependent tuples still refer, is deleted.

Example 20 *Consider the following relational database X consisting of 3 tables* employee, lecturer *and* course*,*

employee		
EmpNo	*Name*	*Salary*
1	*Peter*	*2000*
2	*Mary*	*2100*
3	*Tom*	*1800*

lecturer	
EmpNo	*Topic*
1	*pascal*
1	*prolog*
2	*pascal*
2	*c*

course		
CrsNo	*Topic*	*EmpNo*
cs001	*prolog*	*1*
cs002	*prolog*	*1*
cs003	*pascal*	*1*
cs004	*c*	*2*

and a referential integrity specification requiring the deletion of corresponding lecturer tuples as soon as a resp. employee tuple is deleted, and causing the deletion of corresponding course tuples as soon as a resp. lecturer tuple is deleted, which is expressed in SQL by the following ALTER commands:

```
ALTER TABLE lecturer FOREIGN KEY (EmpNo) REFERENCES employee
ON DELETE CASCADE;
```

```
ALTER TABLE course FOREIGN KEY (EmpNo,Topic) REFERENCES lecturer
ON DELETE CASCADE
```

A first attempt to express corresponding action rules yields:

$$
\begin{array}{ll}
(r_1) & -lect(EmpNo, Topic) \leftarrow -emp(EmpNo, Name, Salary) \\
(r_2) & -crs(CrsNr, Topic, EmpNo) \leftarrow -lect(EmpNo, Topic)
\end{array}
$$

Notice however that such deletion rules of the form $-q(x) \leftarrow -p(x)$ are not admissible action rules since their premise is not an evaluable query. We could try to remedy this problem by adding the 'neutral' positive literal $q(x)$ to the premise yielding $-q(x) \leftarrow q(x) \wedge -p(x)$. But now we have another problem: an 'instable' rule for which there is no stable closure. Therefore, we have to use extended action rules in order to express referential integrity:

$$
\begin{array}{ll}
(r_1) & -lect(EmpNo, Topic) \leftarrow \mathsf{NEvt}(-emp(EmpNo, Name, Salary)) \\
(r_2) & -crs(CrsNr, Topic, EmpNo) \leftarrow \mathsf{NEvt}(-lect(EmpNo, Topic))
\end{array}
$$

If, for instance, Mary quits her job, we would get the following update (using obvious abbreviations, and where $R = \{r_1, r_2\}$):

$$
\begin{array}{lcl}
X' & = & R(\mathsf{Upd}(X, -emp(2, mary, 2100))) \\
 & = & r_2(r_1(X - \{emp(2, mary, 2100)\})) \\
 & = & r_2(X - \{emp(2, mary, 2100),\ lect(2, pascal),\ lect(2, c)\}) \\
 & = & X - \{emp(2, mary, 2100),\ lect(2, pascal),\ lect(2, c),\ crs(cs004, c, 2)\}
\end{array}
$$

While in SQL it is not possible to specify *insertion* rules in order to maintain referential integrity,[8] this is straightforward in our formalism. Let the above set of action rules be extended by adding

$$
(r_3) \quad lect(EmpNo, Topic) \leftarrow \mathsf{NEvt}(crs(CrsNo, Topic, EmpNo))
$$

[8] SQL databases simply issue an error message when a new tuple refering to some non-existing master tuple is to be inserted.

If then, for instance, a new course in logic is established by inserting *crs(cs005,logic,3)*, we get the following update:

$$\begin{aligned} X'' &= R(\mathsf{Upd}(X',\ crs(cs005,logic,3)) \\ &= r_3(X' \cup \{crs(cs005,logic,3)\}) \\ &= X' \cup \{crs(cs005,logic,3),\ lect(3,logic)\} \end{aligned}$$

This example suggests that the automatic maintenance of referential integrity can best be achieved by using resp. symmetric action rules, i.e. a pair consisting of a deletion and an insertion rule, providing for the necessary deletions when a master tuple is deleted and providing for the insertion of a new master tuple (with default assignments) when a new tuple depending on a non-existing master tuple is inserted.

5.7 Conclusion

We have presented a general semantics for action rules in knowledge representation systems. Unlike other work, we stress that action rules can be used not only in relational databases but in all kinds of knowledge bases.

We have also shown that the formalism of active databases is more general than that of deductive databases since the latter can be embedded in the former. In practice, however, one would hardly transform deduction into action rules because this would transform all intensional relations into extensional ones, and the economic advantage of using deduction rules for expressing derived relations implicitly would get lost. Practical systems, therefore, will combine both deduction rules (e.g. for expressing subsumption relationships) and action rules (e.g. for expressing integrity maintenance policies).

It is only natural to consider then, in a next step, communicative actions as an option for active KBs. An active KB may communicate some newly acquired fact to another KB within the communication network it belongs to. E.g., for the supervision of a large technical system several specialized active KBs may periodically run specific diagnostic tests on the resp. hardware components they are connected with, and report the resp. outcomes on certain conditions to a supervisor KB which in turn may inform a human engineer if the incoming messages sufficiently suggest some problem. Such a distributed diagnostic system consists of *cooperative KBs* which will be the topic of a future paper.

Chapter 6

Disjunctive Knowledge and Logic Programming

6.1 Three Fundamental Ambiguities in the Semantics of Disjunctive Logic Programs

We discuss three ambiguities in the semantics of normal disjunctive logic programs, namely:

1. Exclusive or inclusive disjunction ?
2. Which kind of failure ?
3. Local or global application of rules, resp. Disjunction in the Premise ?

6.1.1 Exclusive or Inclusive Disjunction ?

Many proposals on the semantics of disjunctive logic programs, since guided by the idea of minimal models, account only for an exclusive type of disjunction where

$$\{p \vee q\} \vdash -p \vee -q$$

which seems to be an unjustified stipulation.

In $D\boldsymbol{A}$, on the other hand, exclusive disjunction can be defined in a straightforward manner as $p|q := (p \wedge -q) \vee (q \wedge -p)$. This seems preferrable to any exclusive semantics where one cannot express standard disjunctive information being obviously inclusive.

The issue of an inclusive versus exclusive interpretation of disjunction in logic programming was already stressed in [RT88] and [Sak89]. The *possible model semantics* of the latter, which is elaborated in [SI94], seems to come close to the semantics of disjunctive KBs presented here, since – in a way – 'possible models' are produced by $(U\vee)$.

6.1.2 Which Kind of Failure ?

As opposed to definite programs, the notion of failure is no longer clear in disjunctive programs. Various suggestions have been been made for failure definitions, often in the form of 'closed-world assumptions'. The basic idea for the notion of definite failure is the GCWA of [Min82], requiring the failure in all 'epistemic alternatives' the disjunctive KB consists of. In most work following [Min82], however, only minimal sets were admitted as epistemic alternatives, but since failure on the basis of minimal epistemic alternatives is not adequate (see also the previous section), various rather technical *CWA definitions (where * is some string like 'E','EG') have been proposed to remedy this.

In the framework of disjunctive knowledge systems, epistemic alternatives (or possible situation descriptions) are not restricted to minimal ones, and therefore the basic idea of definite failure is adequate. We have also argued above that it seems methodologically preferrable to separate the issue of failure on the basis of disjunctive knowledge from the more complicated problem of failure on the basis of disjunction *and* rules.

6.1.3 Local versus Global Application of Rules

In the case of a disjunctive KB $Y \subseteq L_{\mathrm{KB}}$, a *local closure* of $\langle Y, R\rangle$, corresponding to the local application of rules, can be defined as the set of all informationally minimal $X \in L_{\mathrm{KB}}$ satisfying the rules of R as constraints:

$$\begin{aligned} R_{loc}(Y) &:= \bigcup\{\min R^{\vdash}(X) : X \in Y\} \\ &= \min\{X' \in L_{\mathrm{KB}} : X' \geq X \text{ for some } X \in Y, \text{ such that} \\ &\qquad \text{for all rules } F \leftarrow G \in R,\ X' \vdash F \text{ whenever } X' \vdash G\} \end{aligned}$$

Syntactically, a normal disjunctive logic program can be interpreted either as a KB of $RD\boldsymbol{A}$, or as a KB of $DR\boldsymbol{A}$. In the latter case, rules are local, while in the former case they are global. For instance, the program $\Pi = \{p \vee q, r \leftarrow p, r \leftarrow q\}$ has two readings:

1. As a KB of $DR\boldsymbol{A}$, i.e. as

   $$Y_1 = \{\,\{p, r \leftarrow p, r \leftarrow q\},\ \{q, r \leftarrow p, r \leftarrow q\},\ \{p, q, r \leftarrow p, r \leftarrow q\}\,\}$$

   from which r is inferrable: $Y_1 \vdash r$, since $R(X) \vdash r$, for all $\langle X, R\rangle \in Y_1$.

2. As a KB of $RD\boldsymbol{A}$, i.e. as

   $$\langle Y_2, R_2\rangle \;=\; \langle\{\{p\},\{q\},\{p,q\}\},\ \{r \leftarrow p, r \leftarrow q\}\rangle$$

   from which r cannot be inferred, since $R_2(Y_2) = Y_2$, i.e. no rule from R_2 is applicable. However, using the local closure one obtains

   $$(R_2)_{loc}(Y_2) = \{\{p,r\},\{q,r\},\{p,q,r\}\}$$

   and, as in $DR\boldsymbol{A}$, r can be inferred.

It is conceivable that a disjunctive KB uses both local and global rules, like KBs of $RDR\boldsymbol{A}$.

Observation 27 (Disjunction in the Premise) *The following property,*

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

holds in $DR\boldsymbol{K}$, but not in $RD\boldsymbol{K}$, where $\boldsymbol{K}$ is any KS.

In most proposals on the semantics of disjunctive logic programs,[1] rules are interpreted locally in a model-theoretic fashion, like implications, and therefore DP holds. However, one could argue that a genuine semantics of deduction rules does not imply DP.

[1] See, e.g., [GL91], or [LMR92].

6.2 Extended Disjunctive Logic Programs

Minker and Ruiz, in [MR93], have shown in a systematic way how various semantics for normal disjunctive programs can be extended to allow for strong negation, thus inducing a semantics for extended disjunctive programs (EDLP). Starting with a semantics for normal disjunctive programs in the form of a preferred model operator *Sem*, and somewhat simplifying the presentation,[2] the basic steps are as follows:

- Define the transformation Π' of an EDLP Π eliminating all occurences of strong negation by introducing new predicate symbols, and its inverse $\Pi^{\sim}$, reintroducing the strongly negated predicates. Notice that bot transformations apply also to (sets of) sets of literals, resp. atoms.

- Then the set of preferred (partial) models of an EDLP Π induced by *Sem* is defined as

$$\mathrm{XSem}(\Pi) := (\mathrm{Cons}(\mathrm{Sem}(\Pi')))^{\sim}$$

Notice that this construction corresponds to our definition of $D_{ex}R\boldsymbol{F}$ where also only consistent possible situation descriptions are accepted, thus leading to an explosive semantics which does not work in the case of contradictory information. Therefore, we prefer the non-explosive system $D_{mi}R\boldsymbol{F}$.

It seems that in $D_{ex}R\boldsymbol{F}$ we can emulate the answer set semantics of [GL91] for EDLPs where disjunction is called 'epistemic' and denoted by *or*. Using a special compilation clause for *or*,

$$(U\,or) \qquad \mathsf{Upd}(Y, F \;\; or \;\; G \leftarrow H) := \mathsf{Upd}(Y, F \leftarrow H) \cup \mathsf{Upd}(Y, G \leftarrow H)$$

we have the following conjecture. Let $Y_\Pi = \mathsf{Upd}(\{\emptyset\}, \Pi)$ consist of the 'split collection' of extended logic programs forming Π. Then, $\Pi \vdash l$ under the answer set semantics of [GL91] iff $Y_\Pi \vdash l$ in $D_{ex}R\boldsymbol{F}$.

6.3 Epistemic Logic Programs

Gelfond [1992] has proposed to extend the framework of EDLPs by adding epistemic modal operators in the premise of rules. An *epistemic logic program* (see [BG93]) consists of rules of the form:

$$\bigvee L \leftarrow F_1 \wedge \ldots \wedge F_m \wedge \bigwedge E$$

where $L \subseteq \mathrm{Lit}$, $E \subseteq \mathrm{XLit}$, and the F's are 'subjective literals', i.e. are of the form $\mathsf{B}l$, $-\mathsf{B}l$, $\mathsf{E}l$, or $-\mathsf{E}l$.

Clearly, such rules are special cases of rules in $RD^5_{mi}\boldsymbol{F}$, resp. $D^5_{mi}R\boldsymbol{F}$. An epistemic logic program specifies an epistemic state, called *world view* in [Gel92], as a collection of *answer sets*: $Y \subseteq 2^{\mathrm{Lit}}$ is an epistemic state specified by an epistemic logic program Π if each $X \in Y$ is an answer set of Π^Y, where

$$\Pi^Y := \{\bigvee L \leftarrow \bigwedge E : Y \vdash \bigwedge F_i\}$$

It will be an issue for future work to show how inference from epistemic logic programs in this sense corresponds to inference in $D^5_{mi}R\boldsymbol{F}$.

[2] Minker and Ruiz distinguish between *incoherent* programs, where $XSem(\Pi) = \emptyset$, and *inconsistent* programs, where $XSem(\Pi) = \{\mathrm{Lit}\}$. It is not clear to me, however, what is the difference, semantically ?

6.4 The CWA in Answer Set Semantics

The above account of the CWA reducing the derivability of a strongly negated atom formed with a CWA predicate to the derivability of its weak negation applies to every KS with strong and weak negation in a uniform way. It is, therefore, more general than the definition of the CWA given in [GL90, GL91] for extended logic programs which was modified in [Gel92] in order to account properly for disjunctive information. Thus, in the answer set semantics, there are two different forms of the CWA: one for the definite case, and another one (using an epistemic modal operator) for the disjunctive case. We shall discuss this by means of an example.

In [GL90, GL91] it was proposed to express the CWA for a predicate p by the rule

$$\sim p(x) \leftarrow -p(x)$$

Consider the following example. A member of the scientific council must be both a lecturer and an active researcher. Suppose we know that Peter is a lecturer, $l(P)$, and we also know that Tom is a lecturer or a researcher, $l(T) \vee r(T)$, and for both predicates the CWA holds, then, according to [GL91] we get the following EDLP:

$$\Pi_1 = \begin{cases} l(P) \\ l(T) \vee r(T) \\ \sim l(x) \leftarrow -l(x) \\ \sim r(x) \leftarrow -r(x) \end{cases}$$

which has two *answer sets*,[3] namely $\{l(P),\ \sim r(P),\ l(T),\ \sim r(T)\}$ and $\{l(P),\ \sim r(P),\ r(T),\ \sim l(T)\}$, and consequently, $\sim r(P)$ as well as $\sim l(T) \vee \sim r(T)$ can be inferred, that is, we get the answer **no** not only to the query 'Does Peter belong to the scientific council ?' but also to 'Does Tom belong to the scientific council ?'.

However, as Gelfond argues in [Gel92], the fact that $r(T)$ does not hold in some answer set, does not guarantee that a rational reasoner does not have a reason to belief $r(T)$. Such a reason may be given by the existence of another answer set containing $r(T)$. Gelfond, therefore, proposes to express the CWA for a predicate p in the presence of disjunctive information by means of an epistemic modality in the following way:

$$\sim p(x) \leftarrow -\mathsf{E}p(x)$$

where the premise requires that there is no possible evidence for $p(x)$ in none of the answer sets. We thus obtain the following epistemic logic program as the correct formalization of our problem:

$$\Pi_2 = \begin{cases} l(P) \\ l(T) \vee r(T) \\ \sim l(x) \leftarrow -\mathsf{E}l(x) \\ \sim r(x) \leftarrow -\mathsf{E}r(x) \end{cases}$$

Π_2 has the unique world view

$$\{\{l(P),\ \sim r(P),\ l(T)\},\ \{l(P),\ \sim r(P),\ r(T)\}\}$$

[3] Recall that $S \subseteq \mathrm{Lit}$ is an *answer set* of an EDLP Π if $S \in \alpha(\Pi^S)$ where Π^S is the Gelfond-Lifschitz transformation eliminating negation-as-failure from Π, and $\alpha(\Pi)$ collects all minimal literal closures of Π.

yielding the conclusion that $\sim r(P)$, but not that $\sim l(T) \vee \sim r(T)$, and hence as desired only the query 'Does Peter belong to the scientific council ?' gets the answer **no**, while the same query with respect to Tom gets the answer **unknown** now.

This is exactly the same result as in the corresponding disjunctive fact base obtained by stipulating the CWA for the predicates l and r, and compiling the facts that Peter is a lecturer and that Tom is a lecturer or a researcher:

$$Y \;=\; \begin{cases} \mathrm{CWA} = \{l, r\} \\ \mathsf{Upd}(0, l(P) \wedge (l(T) \vee r(T))) \\ = \;\{\{l(P),\, l(T)\},\, \{l(P),\, r(T)\},\, \{l(P),\, l(T),\, r(T)\}\} \end{cases}$$

We get the following closure:

$$CWA(Y) \;=\; \{\{l(P),\, l(T),\, \sim r(P)\},\, \{l(P),\, r(T),\, \sim r(P)\},\, \{l(P),\, l(T),\, r(T),\, \sim r(P)\}$$

yielding the same answers as above to the queries whether Peter and Tom belong to the scientific council.

6.5 Conclusion

The framework of normal logic programs can be extended with respect to at least four big logical issues: explicit negative information, disjunctive information, modal queries (resp. subjective knowledge), and conditional queries. In all these extensions there are different design options how to model the interaction of negations and the other operators. And it is not obvious at all what is the 'right' generalization of negation-as-failure in these settings. We hope, however, that the proposed framework of vivid knowledge representation systems helps to realize and evaluate the different options and modelings.

Chapter 7

Solving the Steamroller, and Other Puzzles, with Extended Disjunctive Logic Programming

Abstract: We show how to solve the classical ATP benchmark test problem *Schubert's Steamroller*, and other puzzles, in the nonclassical framework of extended disjunctive logic programming (EDLP) where neither Contraposition nor Reasoning by Cases are valid principles of inference. While the 'Steamroller' can be solved using only Detachment and Disjunctive Syllogism, another puzzle, 'Who killed aunt Agatha ?', needs in addition a local form of Contraposition, resp. Reasoning by Cases, which is achieved by declaring certain predicates as *exact* in the sense of [Kör66]. Besides being able to solve certain classical ATP problems in a monotonic reasoning mode, EDLP can as well treat commonsense reasoning problems by employing its intrinsic nonmonotonic inference capabilities. This will be illustrated by means of a nonmonotonic variant of the 'Steamroller' problem. EDLP thus proves itself as a powerful general-purpose AI reasoning system.

7.1 Introduction

Logic programs with both strong negation and disjunction are called *extended disjunctive logic programs (EDLP)*.[1] Despite their comparable expressiveness they do not lead to classical first order logic. In EDLPs, neither the *Tertium Non Datur* (resp. Reasoning by Cases) nor the *Contraposition* principle hold. Our conjecture is, however, that these classical principles in their full generality are not needed in many cases of natural inferences, especially in knowledge-based reasoning (as opposed to theoretical reasoning). The 'Steamroller' is an example for this. Although it has been formulated as a challenging benchmark test for classical *automated theorem proving (ATP)* systems by L. Schubert,[2] it can be naturally represented and efficiently solved as an EDLP. In contrast to ATP systems using full classical first order logic, however, EDLPs are evaluated by means of only two logical principles: *Detachment*, i.e. rule application, and some form of the *Disjunctive Syllogism*. They seem to be more efficient therefore.

We do not suggest that EDLPs, resp. the underlying framework of vivid knowledge representation systems, are an alternative to classical ATP systems. Rather we would like to

[1] See, e.g., [GL91], or [MR93].

[2] For a comprehensive description of it including historical remarks see [Sti86].

argue that purely theoretical, notably mathematical, reasoning should be distinguished from practical reasoning such as embodied in databased and knowledge-based inferences. While classical ATP seems appropriate for the former, vivid knowledge representation systems, such as extended logic programming, seem to be more appropriate for the latter.

The main purpose of the paper is not to present and discuss a new theory (although we will have to do that to some extent), but rather to demonstrate the subtle interaction between theoretical modeling and practical KR requirements arising from the treatment of relevant and, to some extent, generic examples (such like the 'Steamroller' and 'Aunt Agatha'). We apologize to those readers looking in vain for more explanation and justification of our formalism. But we believe that the thourough discussion of non-trivial examples is interesting and substantial by itself.

The structure of the paper is as follows: in section 2 we will present some basic concepts of the theory of knowledge representation proposed in [Wag94a], notably the concept of a rule knowledge base and its closure semantics. A new general construction for disjunctive knowledge bases is presented in section 3. It is the basis for showing then in section 4 how to represent and solve the 'Steamroller' as a monotonic EDLP. In section 5, a nonmonotonic version of the 'Steamroller' where certain rules express defaults, and thus may have exceptions, is discussed. Finally, the concept of exact predicates and the related pinciple of local Reasoning by Cases, resp. local Contraposition, is used to solve the 'Aunt Agatha' problem.

The main contributions of the paper are: a) a short introduction of conditionals to the language of rules in section 2.3, b) the new concept of a *disjunctive KRS*[3] in section 3, and c) the demonstration how to treat puzzles requiring the language of first order logic as EDLPs in the framework of vivid knowledge representation systems.

7.2 Introducing a First-Degree Conditional

In knowledge-based reasoning, universal quantification is always restricted, i.e. we never have expressions like $\forall x(q(x))$ as queries or inputs, but only *extensional subsumption* formulas like $\forall x(p(x) \supset q(x))$ as queries, or *intensional subsumption* formulas like $\forall x(p(x) \to q(x))$ as queries and inputs. We shall omit the quantifier symbol $\forall$ in such conditionals and interpret all free variables of the premise as implicitly universally quantified.

Extensional subsumption is expressed by means of a 'material implication'-style operator $\supset$ which can be added to the query language of any KRS according to the following definition:

$$X \vdash F \supset G \quad :\Longleftrightarrow \quad X \vdash \bigwedge_{\{\sigma : X \vdash F\sigma\}} G\sigma$$

where $F, G \in L^x_{\text{Query}}$ such that $\mathsf{Free}(F) \supseteq \mathsf{Free}(G)$.[4] For an example see the rule $(F2)$ in 5.2.

Intensionl subsumption can be expressed by means of a conditional operator $\to$ which can be added both to the input and to the query language of any rule knowledge representation system. Let $R\boldsymbol{K}$ be the rule-based extension of a KRS $\boldsymbol{K}$. Then the *conditional extension* $\vec{R}\boldsymbol{K}$ of $R\boldsymbol{K}$ is defined by 1) the input language

$$\vec{L}_{\text{Input}} := \{F \to G : F \in L^x_{\text{Query}} \;\&\; G \in L^x_{\text{Input}},\ \text{such that } \mathsf{Free}(G) \subseteq \mathsf{Free}(F)\}$$

[3]A preliminary report on disjunctive KRSs has been presented in [Wag93b]. A more elaborate report containing some highlighting theoretical results is in preparation.

[4]Notice that this definition contains the special case where F and G are ground, and hence $X \vdash F \supset G$ iff $X \vdash -F \vee G$.

2) the update opertion

$$\mathsf{Upd}(\langle X,R\rangle,F\to G):=\begin{cases}\langle\mathsf{Upd}(X,G),R\rangle & \text{if } F=1\\ \langle X,R\cup\{G\leftarrow F\}\rangle & \text{otherwise}\end{cases}$$

3) the query language

$$\vec{L}_{\text{Query}}:=\{F\to G: F\in L^x_{\text{Input}}\ \ \&\ \ G\in L^x_{\text{Query}},\ \text{such that } \mathsf{Free}(G)\subseteq\mathsf{Free}(F)\}$$

4) the inference relation

$$\langle X,R\rangle\vdash F\to G\quad :\Longleftrightarrow\quad \langle\mathsf{Upd}(X,F\nu),\ R\rangle\vdash G\nu$$

where ν is a special substitution replacing each free variable of F with a *fresh* constant symbol.

We shall not consider negated conditionals in this paper.

Example 21 *The following $R\vec{R}\boldsymbol{F}$ rule, $r=[\sim b(x)\to r(x)]\leftarrow l(P,x)$, may express that every woman Peter likes has red hair if she is not blonde. If applied to the $\vec{R}\boldsymbol{F}$ rule knowledge base*

$$\langle X_1,R_1\rangle=\langle\{l(P,S),\sim b(S)\},\ \{l(P,x)\leftarrow b(x)\}\rangle,$$

one obtains

$$r(\langle X_1,R_1\rangle)=\langle\{l(P,S),\sim b(S)\},\ \{l(P,x)\leftarrow b(x),\ r(S)\leftarrow\sim b(S)\}\rangle$$

7.2.1 Exact Predicates

Observation 28 *In consistent KBs of $D\boldsymbol{K}$, weak and strong negation coincide for exact predicates. Let* KB *be consistent, p be exact, t be any ground term from its domain, then*

$$\text{KB}\vdash\sim p(t)\quad \textit{iff}\quad \text{KB}\vdash -p(t)$$

Observation 29 (Weak Contraposition) *In $DR\boldsymbol{K}$ there is a weak form of contraposition: if Y is consistent, then*

$$Y\vdash\sim F\ \ \&\ \ Y\vdash G\to F\ \textit{ implies }\ Y\vdash -G$$

Therefore, if $q\in EXACT$, and $p(x)\leftarrow q(x)\in\bigcap\{[R]:\langle X,R\rangle\in Y\}$, contraposition may be used in $DR\boldsymbol{K}$, $Y\vdash\sim p(t)\ \Rightarrow\ Y\vdash\sim q(t)$.

7.3 Schubert's Steamroller

In Schubert's Steamroller all universal quantifications are restricted, and hence can be expressed by open conditionals, resp. rules. The representation as an EDLP is much more natural (and readable) than the standard clausal representation used in classical ATP systems. In order to solve the 'Steamroller' neither Reasoning by Cases nor Contraposition is needed.

7.3.1 Representation

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them:[5]

$$\begin{array}{ll} (S1) & a(x) \leftarrow w(x) \vee f(x) \vee b(x) \vee c(x) \vee s(x) \\ (S2) & w(w_1) \wedge f(f_1) \wedge b(b_1) \wedge c(c_1) \wedge s(s_1) \end{array}$$

Also, there are some grains, and grains are plants:

$$\begin{array}{ll} (S3) & g(g_1) \\ (S4) & p(x) \leftarrow g(x) \end{array}$$

Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves:

$$\begin{array}{ll} (S5) & sm(x,y) \leftarrow b(y) \wedge (c(x) \vee s(x)) \\ (S6) & sm(x,y) \leftarrow b(x) \wedge f(y) \\ (S7) & sm(x,y) \leftarrow f(x) \wedge w(y) \\ (S8) & sm(x,y) \leftarrow sm(x,z) \wedge sm(z,y) \end{array}$$

Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails:

$$\begin{array}{lrl} (S9) & \sim l(x,y) & \leftarrow w(x) \wedge (f(y) \vee g(y)) \\ (S10) & \sim l(x,y) & \leftarrow b(x) \wedge s(y) \\ (S11) & l(x,y) & \leftarrow b(x) \wedge c(y) \end{array}$$

Caterpillars and snails like to eat some plants:

$$\begin{array}{ll} (S12) & l(x, p_2(x)) \wedge p(p_2(x)) \leftarrow c(x) \\ (S13) & l(x, p_3(x)) \wedge p(p_3(x)) \leftarrow s(x) \end{array}$$

Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants, yielding the following (nested) rule:

$$(S14) \qquad [p(y) \rightarrow l(x,y)] \vee [a(z) \wedge sm(z,x) \wedge p(u) \wedge l(z,u) \rightarrow l(x,z)] \ \leftarrow \ a(x)$$

Conclusion: there is an animal that likes to eat a grain-eating animal,

$$\exists x \exists y \exists z : a(x) \wedge a(y) \wedge g(z) \wedge l(x,y) \wedge l(y,z).$$

The $RD\vec{R}\boldsymbol{F}$ rule[6] $(S14)$ can be transformed into the (not quite equivalent) $RD\boldsymbol{F}$ rule

$$(S14') \qquad l(x,y) \vee l(x,z) \ \leftarrow \ a(x) \wedge p(y) \wedge a(z) \wedge sm(z,x) \wedge p(u) \wedge l(z,u)$$

The difference between them, however, is not significant for the solution of the 'Steamroller'.

---

[5] Existential information is represented by means of 'null values', resp. skolem functions. For instance, $\exists x w(x)$ is transformed into $w(n_k^w)$ where n_k^w is a newly introduced null value associated with the domain of the predicate w and enumerated by $k = 1, 2, 3, \ldots$. In order to simplify notation we write w_k instead of n_k^w.

[6] F denotes the system of fact bases, $\vec{R}F$ denotes its rule-based extension (i.e. extended logic programs) allowing for conditional input formulas, $D\vec{R}F$ denotes the disjunctive extension of $\vec{R}F$ where KBs consist of a collection of extended logic programs, and finally $RD\vec{R}F$ allows to add rules operating on $D\vec{R}F$ such as $(S14)$.

7.3.2 Sketch of solution

We shall present a solution in $RD\vec{R}\boldsymbol{F}$, using $(S14)$, rather than with $(S14')$ in $RD\boldsymbol{F}$, because it leads to less 'disjunctiveness' and hence seems to be more efficient. In $RD\vec{R}\boldsymbol{F}$, the solution is obtained from the rule knowledge base $\langle Y, R\rangle$ where

$$Y = \{\langle\{S2, S3\}, \emptyset\rangle\}$$

and $R = \{S1, S4, S5, \ldots, S13, S14\}$. The rules of R are successively applied in the bottom-up computation of $R(Y)$. In the case of branching into different possible situation descriptions, i.e. when evaluating $(S14)$, their consistency has to be checked in order to apply the Disjunctive Syllogism, i.e. discarding the inconsistent ones.

The 5 disjunctions on what each animal likes to eat yield, in the first calculation step, 2^5 possible combinations, i.e. 32 different rule knowledge bases as elements of $R(Y)$. Each of them contains in addition to the definite information common to all of them, including all facts involving the predicates a, p, g, sm, among others:

$$p(p_2(c_1)),\ p(p_3(s_1)),\ sm(c_1, w_1),\ sm(s_1, w_1),\ sm(b_1, w_1),$$

and some facts invloving l and $\sim l$, such as, among others:

$$l(b_1, c_1),\ {\sim}l(w_1, g_1),\ {\sim}l(b_1, s_1),$$

a certain combination of the *likes-plants* resp. *likes-animals* rules such as

$$\begin{array}{l} l(b_1, x)) \leftarrow p(x) \\ l(w_1, x) \leftarrow a(x) \wedge sm(x, w_1) \wedge p(y) \wedge l(x, y) \\ l(f_1, x)) \leftarrow p(x) \\ l(c_1, x)) \leftarrow p(x) \\ l(s_1, x)) \leftarrow p(x) \\ \ldots \end{array}$$

The number of 2^5 such combinations is reduced to 2^3 by the Disjunctive Syllogism, namely:

1. Birds definitely like to eat all plants since they dislike snails.
2. Wolves definitely like all smaller animals that like to eat some plants (i.e. caterpillars, snails and birds) since they dislike grain.

Consequently all 8 possible situation descriptions contain the facts that a bird likes to eat a grain, $l(b_1, g_1)$, and that a wolf likes to eat a bird, $l(w_1, b_1)$. Hence the answer: yes, there is an animal that likes to eat a grain-eating animal (and, moreover, both the species of this animal and of its prey are known: it is a wolf and a bird).

7.3.3 The Usual Interpretation

In the literature on the problem, the predicate *smaller* has not been transitively closed, and therefore it has not been concluded that birds are smaller than wolves (blocking the conclusion that wolves like to eat birds).[7] Instead, it has been concluded (by an instance of Reasoning by Cases, and the Disjunctive Syllogism) that, since foxes either like to eat all plants or not and wolves like all smaller plant-eating animals but not foxes, foxes do not

[7] We wonder why, since it is certainly part of the natural meaning of *smaller* that it is transitive.

like to eat all plants, implying (again by DS) that they like all smaller animals that like to eat some plants, in particular birds, and consequently the answer: yes, namely $l(f_1, b_1)$.

This kind of solution can also be obtained in our framework, by deleting the transitive closure rule for *smaller*, $(S8)$, and adding an appropriate instance of the Tertium Non Datur, e.g. $l(f_1, g_1) \vee \sim l(f_1, g_1) \leftarrow 1$, which is not automatically available in a disjunctive KB but only if the resp. predicate is declared to be *exact* as explained in section 6. However, in the case of the predicate *likes* one can hardly assume it to be exact (i.e. without truth value gaps), since it is an empirical predicate which may be neither verifiable nor falsifiable for specific instances.[8]

7.4 The Nonmonotonic Steamroller

We will now interpret the statements *S9–S13* about animals liking or disliking to eat other animals or plants as a kind of default: *normally* they do so. In many cases, this seems to be more natural for empirical predicates, such as *likes*, than their strict interpretation.

First notice that default rules cannot be expressed in standard first order logic, and therefore classical ATP systems cannot handle such commonsense puzzles. There are at least two possibilities how to obtain default reasoning in extended logic programming: 1) by means of *exception handling*, and 2) by means of interpreting rules as *defeasible* using the semantical principles of *neutralization* and *preemption*. While the exception-handling approach requires the addition of an exception check to the premise of a rule, the syntactic form of a rule needs not to be changed in the defeasible reasoning approach (which may require a priority ordering of the rules, however).[9] In the sequel we will focus on the exception handling method proposed in [BG93]. It would be interesting, however, to compare both approaches in terms of expressiveness and efficiency.

7.4.1 Dealing with Exceptions

In [BG93], a default rule r like *'Normally F if G'* is expressed by a program rule with an additional premise requiring that individuals (resp. tuples) to which the rule is applicable are not known to be 'weak exceptions' for which the rule does not hold (recorded by means of a predicate ex_r), or 'strong exceptions' for which even the contrary holds:

$$(r) \quad F(x) \leftarrow G(x) \wedge -ex_r(x) \wedge -\sim F(x)$$

According to this representation method we obtain the following modifications of *S9–S13*:

$$\begin{array}{lrcl}
(S9') & \sim l(x,y) & \leftarrow & w(x) \wedge (f(y) \vee g(y)) \wedge -ex_9(x,y) \wedge -l(x,y) \\
(S10') & \sim l(x,y) & \leftarrow & b(x) \wedge s(y) \wedge -ex_{10}(x,y) \wedge -l(x,y) \\
(S11') & l(x,y) & \leftarrow & b(x) \wedge c(y) \wedge -ex_{11}(x,y) \wedge -\sim l(x,y) \\
(S12') & l(x,p_2(x)) \wedge p(p_2(x)) & \leftarrow & c(x) \wedge -ex_{12}(x) \wedge (p(y) \rightarrow -\sim l(x,y)) \\
(S13') & l(x,p_3(x)) \wedge p(p_3(x)) & \leftarrow & s(x) \wedge -ex_{13}(x) \wedge (p(y) \rightarrow -\sim l(x,y))
\end{array}$$

It is easy to see that with $(S9)$–$(S13)$ replaced by their resp. exception handling modification $(S9')$–$(S13')$ we still get the same positive answer as before simply because the available null values are not known to be exceptional. Only by introducing explicit exceptions we can

[8] We doubt that wilderness biologists would give a yes-or-no answer to the question whether all foxes eat grain or not.

[9] Defeasible reasoning with logic programming rules is based on a non-standard semantics of rule application (resp. *Detachment*) such as proposed, e.g., in [Nut88], or [Wag93a].

change the picture. Therefore, we add to the above story that *there is a special bird, called Tweety, for which the rule* $(S10')$ *does not hold with respect to any snail*, i.e.

$(T1)$ $b(T)$
$(T2)$ $ex_{10}(T, x) \leftarrow s(x)$

Also we introduce a strong exception to $(S11')$, namely *Opus, a special bird which dislikes caterpillars:*

$(O1)$ $b(O)$
$(O2)$ $\sim l(O, x) \leftarrow c(x)$

7.4.2 Reasoning with Unknown Information

As an example of reasoning on the basis of unknown information we add that *there is a special fox, called Foxy, which likes to eat all smaller animals for which it is (definitely) not known that they eat other animals.*

$(F1)$ $f(F)$
$(F2)$ $l(F, x) \leftarrow a(x) \wedge sm(x, F) \wedge (l(x, y) \supset -a(y))$

It is easy to see that, according to $(F2)$, Foxy would like to eat Opus since there is no evidence that Opus likes to eat any animals, and as Opus dislikes caterpillars and therefore likes all plants, including grain, a second solution to the query asking for all animals that like to eat a grain-eating animal is obtained.

7.4.3 The Nonmonotonic Steamroller

We propose the following nonmonotonic reasoning problem as a benchmark test for automated reasoning systems:

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Specifically, Tweety and Opus are birds, and Foxy is a fox. Also, there are some grains, and grains are plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do normally not like to eat foxes or grains. Birds normally like to eat caterpillars; however, Opus dislikes them. Except Tweety, birds normally do not like snails. Caterpillars and snails like to eat some plants (under normal conditions). Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Foxy likes to eat all smaller animals for which it is not known that they eat other animals. Query: which animals like to eat a grain-eating animal ?

In $RD\vec{R}\boldsymbol{F}$, this yields the KB $\langle Y, R\rangle$ with

$$Y = \{\langle\{S2, S3, T1, O1, F1\},\ \emptyset\rangle\}$$

and $R = \{S1, S4, S5, S6, S7, S8, S9', S10', S11', S12', S13', S14, T2, O2, F2\}$, from which we can conclude that Opus likes to eat some grain, and that both a wolf and Foxy like to eat Opus:

$$R(Y) \vdash l(O, g_1) \wedge l(w_1, O) \wedge l(F, O).$$

7.5 Who killed Aunt Agatha ?

The following problem, due to L. Schubert, is borrowed from [Pel86]. Its solution in $DR\boldsymbol{F}$ is obtained by using (in addition to the basic principles Detachment and Disjunctive Syllogism) the posibility to declare certain predicates as exact, thus enabling local Reasoning by Cases, resp. local Contraposition.

7.5.1 Representation

Someone who lives in Dreadsbury Mansion, killed aunt Agatha. Agatha, the butler, and Charles live in Dreadsbury Mansion, and are the only people who live therein:[10]

$$(A1) \quad k(A,A) \vee k(B,A) \vee k(C,A)$$

A killer always hates her victim, and is never richer than her victim:

$$(A2) \quad h(x,y) \wedge {\sim} r(x,y) \leftarrow k(x,y)$$

Charles hates no one that aunt Agatha hates:

$$(A3) \quad {\sim} h(C,x) \leftarrow h(A,x)$$

Agatha hates everyone except the butler, and no one the butler does not hate:[11]

$$\begin{array}{lrl} (A4) & h(A,x) & \leftarrow {\sim} e(x,B) \\ (A5) & {\sim} h(A,x) & \leftarrow {\sim} h(B,x) \end{array}$$

The butler hates everyone not richer than aunt Agatha:

$$(A6) \quad h(B,x) \leftarrow {\sim} r(x,A)$$

No one [in Dreadsbury Mansion] hates everyone:

$$(A7) \quad {\sim} h(A,h_1(A)) \wedge {\sim} h(B,h_1(B)) \wedge {\sim} h(C,h_1(C))$$

Agatha is not the butler:

$$(A8) \quad {\sim} e(A,B)$$

Who killed aunt Agatha: $k(x,A)$?

7.5.2 Sketch of Solution

As opposed to the predicate *hates* which may be undetermined in certain cases, it seems plausible to declare the predicates *kills*, *richer* and *equals* as exact, i.e. assuming that they do not have truth value gaps, or in other words, that for all ground instances of them it can be determined whether they are true or false:

$$\mathrm{EXACT}(k,r,e)$$

While in $DR\boldsymbol{F}$ the solution is found by 'brute force' bottom-up computation, we will construct here only the essential reasoning steps using the weak contraposition principle. Two

[10] We reduce this information, namely $k(k_1,A), lDM(k_1), lDM(A), lDM(B), lDM(C)$, together with CWA $= \{lDM\}$, to the disjunction $k(A,A) \vee k(B,A) \vee k(C,A)$.

[11] Notice that we use e as an equality predicate. In the case of $(A5)$ we use the contraposition of Schubert's original formulation 'The butler hates everyone Agatha hates.'

arguments can be found why neither Charles nor the butler killed Agatha, and hence, by the Disjunctive Syllogism, she must have killed herself.

From $(A4)$ and $(A8)$ we get immediately that Agatha hates herself, consequently by $(A3)$, Charles does not hate Agatha, and hence with the contraposition of $(A2)$, he did not kill her, $\sim k(C, A)$.

From $(A5)$ and $(A7)$ we obtain $\sim h(A, h_1(B))$. Using the contraposition of $(A4)$ we get $e(h_1(B), B)$, and hence from $(A7)$ by simple equality reasoning, $\sim h(B, B)$. By the contraposition of $(A6)$, this implies $r(B, A)$, and consequently, by the contraposition of $(A2)$, $\sim k(B, A)$.

7.6 Conclusion

We have shown by means of some non-trivial examples that extended disjunctive logic programs, resp. vivid rule knowledge bases, can be used to express and solve puzzles which require the language of first order logic. We believe that the rule language of rule-based VKRSs is more natural than the language of standard clauses used in ATP systems. We also believe, and hope to substantiate this belief by experimental results in the final version of this paper, that extended logic programmig, since non-classical, is in many cases more efficient than classical ATP systems.

In addition to these advantages, extended logic programming and VKRSs offer a flexible and powerful framework for nonmonotonic forms of commonsense reasoning as we have illustrated here, and as is also argued in [BG93].

Chapter 8

Contraction and Revision in Vivid Knowledge Systems

A vivid knowledge system (VKS) has, in addition to a standard negation '$\sim$', a second negation in the style of negation-as-failure, called *weak negation*, denoted by '$-$'. In a VKS a contraction operation is definable as updating by weakly negated inputs: $\mathrm{KB} - F := \mathsf{Upd}(\mathrm{KB}, -F)$. Thus, contraction is a particular form of update. Recency-preferring revision can be defined according to $\mathrm{KB} * F := \mathsf{Upd}(\mathsf{Upd}(\mathrm{KB}, -\sim F), F)$.[1]

We present appropriate modifications of the rationality postulates for contraction and revision and compare them to the original ones proposed in the AGM theory[2] and by Hansson [1991]. We also discuss the application of our notions to several VKSs, including fact bases, disjunctive fact bases and their rule-based extensions.

8.1 Introduction

In contrast to standard logics, where dynamic aspects of reasoning are largely neglected, belief change operations, such as update, contraction and revision, are essential components of KSs. The term 'belief revision' is used in at least two ways: in its broader sense belief revision subsumes any systematic approach to the question of how to change a KB when new information comes in, and in its narrower sense it denotes the specific change operation axiomatised by the AGM postulates. In [GR 92] the goal of consistency maintenance is considered the dominating motive for belief revision. We would like to point out that there are reasonable systems, however, not maintaining consistency but nevertheless employing a revision operation.[3]

The main principles for belief change operations, according to Gärdenfors and Rott, are:

1. The beliefs in a KB should be kept consistent (whenever possible).
2. The amount of information lost in a belief change should be kept minimal.
3. In so far as some beliefs are considered more important than others, one should retract the least important ones.

[1] This definition of revision in terms of contraction is due to Levi [1977].
[2] Called after its originators: C. Alchourrón, P. Gärdenfors and D. Makinson. See [Gär88].
[3] E.g., [CT92].

The first principle is called *minimal inconsistency* here, and the second one *minimal change.* Notice that the formulation "whenever possible" in the first principle can be interpreted in different ways. In the AGM theory, the only case where consistency cannot be avoided is when the input itself is inconsistent. In another approach (proposed by Pries [1989]), the principle of minimal inconsistency means that the beliefs in a KB should be kept 'as consistent as possible' in the sense that only the minimally inconsistent 'possible worlds' are accepted, and if there are only inconsistent 'possible worlds' left, no trivial inferences are drawn.

Gärdenfors and Rott also include the principle that KBs should be deductively closed, because this is essential for the formulation of the AGM theory. We will not adopt this principle, since it seems to be inappropriate for KSs where KBs are always finite objects. Therefore, we will rather be concerned with change operations for 'belief bases' instead of infinite 'belief sets', as KBs are called in the AGM terminology. Hansson [1991] has proposed postulates for belief bases. The main difference to the AGM postulates seems to be that Hansson does not require recovery which indeed fails in disjunctive KSs (see below).

In summary, our approach differs from the AGM approach (and related ones) in several respects:

1. KBs are not deductively closed.

2. Contraction is more fundamental than revision. In a VKS, contraction is nothing else as weak negation on the left, i.e. the negative counterpart of ampliative updating,

3. We prefer *direct* constructive notions (such as *strong negation* '$\sim$') to indirect and nonconstructive ones (such as classical negation '$\neg$', or $K \bot F$, or $K_\bot$).

4. The preferred form of a KB is not a set of formulas, but rather some form of 'compiled' information (since every input formula can be compiled, this is no loss of generality). Contraction in such KBs is syntax-independent: e.g. $\lceil p \wedge q \rceil - p = \lceil p, q \rceil - p$. Also, KBs are not compared by means of set inclusion, $\subseteq$, but rather by means of an informational ordering $\leq$.

5. The basic pieces of information in a KS are atoms, resp. literals, and not arbitrary formulas. We are, therefore, most interested in the formalization of contraction and revision for atoms, resp. literals, whereas disjunctions and conditionals may be included in the picture as optional extensions in a second step.

6. Vacuity and Recovery do not hold in general. In many cases, however, they hold for literals.

7. Consistency has not to be maintained under all circumstances in a VKS. Paraconsistency (e.g. in the form of minimal inconsistency) is more realistic. This implies that our logics do not contain the concept of absurdity (expressed by $\bot$ and $K_\bot$) as a fundamental ingredient.

8. Strong negation does not satisfy the inference rule of (Negation as Inconsistency). Therefore, it is not possible to express that F is inconsistent with KB by the condition $\mathrm{KB} \vdash \sim F$. This affects the formulation of the resp. proviso of several of the rationality postulates for revision.

We use the following abbreviations: $\mathrm{KB}+F := \mathsf{Upd}(\mathrm{KB}, F)$ for expansion, and $\mathrm{KB}-F := \mathsf{Upd}(\mathrm{KB}, -F)$ for contraction.

Definition 18 (Entailment) *For $F \in L_{\text{Input}}$ and $G \in L_{\text{Query}}$ define*

$$F \vdash G :\Longleftrightarrow \mathsf{Upd}(\text{KB}, F) \vdash G \textit{ for any } \text{KB} \in L_{\text{KB}}$$

Definition 19 (Answer Equivalence) KB_1 *and* KB_2 *are* answer equivalent, *symbolically* $\text{KB}_1 \simeq \text{KB}_2$, *if they produce the same answers at the moment, i.e. if for any query formula F,*

$$\mathsf{Ans}(\text{KB}_1, F) = \mathsf{Ans}(\text{KB}_2, F)$$

Definition 20 (Update Equivalence) KB_1 *and* KB_2 *are* update equivalent, *symbolically* $\text{KB}_1 \cong \text{KB}_2$, *if they produce the same answers in all circumstances, i.e. if for any input formula F,*

$$\mathsf{Upd}(\text{KB}_1, F) \simeq \mathsf{Upd}(\text{KB}_2, F)$$

8.2 Contraction

We present a list of rationality postulates for contraction in KSs. They are appropriate modifications of postulates originally proposed in [Gär88] as an axiomatization of contraction for belief sets. In our direct approach, however, these postulates are not axioms, but rather expressing interesting properties the contraction operation defined in a KS may or may not have.

(Success)	$\not\vdash F \;\Rightarrow\; \text{KB} - F \vdash -F$
(Inclusion)	If F is ampliative, then $\quad \text{KB} - F \leq \text{KB}$
(Vacuity)	$\text{KB} \vdash -F \;\Rightarrow\; \text{KB} - F \cong \text{KB}$
(Extensionality)	$F \vdash G \;\&\; G \vdash F \;\Rightarrow\; \text{KB} - F \cong \text{KB} - G$
(Recovery)	$\text{KB} \vdash F \;\Rightarrow\; (\text{KB} - F) + F \cong \text{KB}$
(Conjunction 1)	$C(\text{KB} - F) \cap C(\text{KB} - G) \subseteq C(\text{KB} - (F \wedge G))$
(Conjunction 2)	$\text{KB} - (F \wedge G) \not\vdash F \;\Rightarrow\; \text{KB} - (F \wedge G) \;\leq\; \text{KB} - G$

Notice that in our formulation of Vacuity and Success we have used $\text{KB} \vdash -F$ instead of the original $\text{KB} \not\vdash F$. Because of $\text{KB} \vdash -F \;\Rightarrow\; \text{KB} \not\vdash F$, this strengthens the original formulation.

While Recovery is a condition on the interaction between contraction and expansion, the postulates of success, vacuity and extensionality describe properties of the update operation.

8.2.1 Contraction in Fact Bases

Claim 21 *In* $\boldsymbol{F}$, *where only literals, resp. extended literals are admitted as input, the contraction operation '$-$' satisfies inclusion, vacuity, success, relevance, extensionality and recovery.*

8.2.2 Contraction in Disjunctive Data Bases

Observation 30 (Failure of Vacuity) *Vacuity fails in many cases of compound formulas:*

$$\{\{p\}\} \vdash -(p \wedge q), \quad \textit{but} \quad \{\{p\}\} - p \wedge q = \{\{\}, \{p\}\} \not\cong \{\{p\}\}$$

Observation 31 (Failure of Recovery) *Recovery fails in the case of disjunctions:*

$$\{\{p,q\}\} - p \vee q + p \vee q = \{\emptyset\} + p \vee q = \{\{p\},\{q\},\{p,q\}\} \not\geq \{\{p,q\}\}$$

Claim 22 *In $D\boldsymbol{A}$, the contraction operation '$-$' satisfies inclusion, atomic vacuity, success, relevance, extensionality and atomic recovery.*

8.2.3 Contraction in Extended Deductive Databases

Extended deductive databases (XDBs) correspond to rule knowledge bases of $R\boldsymbol{F}$.

Updating by a weakly negated ground literal in an extended deductive database is defined as follows:

Definition 21 (Literal Contraction) *Let l be a ground literal. Then $\mathsf{Upd}(\Pi, -l)$ denotes the XDB obtained from Π by making the following changes:*

1. *If Π contains l as a fact (resp. a rule $l \leftarrow F$ such that $\Pi \vdash F$), then l (resp. $l \leftarrow F$) is deleted from Π.*
2. *If $l = p(c)$, resp. $l = \sim p(c)$, and Π contains the rule $p(x) \leftarrow F(x)$, resp. $\sim p(x) \leftarrow F(x)$, such that $\Pi \vdash F(c)$, then this rule is replaced by $p(x) \leftarrow x \neq c \wedge F(x)$, resp. $\sim p(x) \leftarrow x \neq c \wedge F(x)$.*

Example 22 *We can now apply this contraction operation to an example given in [GR 92]. Let*

$$\Pi = \{p,\ q,\ r \leftarrow q,\ s \leftarrow p \wedge r\}$$

According to Gärdenfors and Rott, it is not clear which element of Π has to be given up in order to obtain the contraction $\Pi - s$. According to our approach, however, this is clear: only the deletion of the rule $s \leftarrow p \wedge r$ leads to an informationally minimal change. The deletion of q, which is supposed to be an equally good candidate according to Gärdenfors and Rott, violates the minimal change principle (-2):

$$C^l(\{p,\ r \leftarrow q,\ s \leftarrow p \wedge r\}) = \{p\} \subset C^l(\{p,\ q,\ r \leftarrow q\}) = \{p,q,r\}$$

Example 23 (View Updates) *In the literature on 'view updates' in deductive databases it is common to ignore the minimal change principle in favour of the (hardly justifiable) idea that only facts should be changed but not rules. For instance, in order to 'delete' $q(c)$ from the database*

$$\Pi = \{p(c),\ r(d),\ q(x) \leftarrow p(x) \wedge r(d),\ s(x) \leftarrow p(x) \vee r(x)\}$$

two possibilities are considered: in the first one, $p(c)$ is deleted, and in the second one, $r(d)$ is deleted. Both changes are not minimal: clearly,

$$\Pi - \{p(c)\} < \mathsf{Upd}(\Pi, -q(c))$$

as well as

$$\Pi - \{r(d)\} < \mathsf{Upd}(\Pi, -q(c))$$

8.3 Consolidation and Revision

While the axiomatization of revision in the AGM theory is inspired by the kind of theory revision investigated in the philosophy of science, our prime concern is revision in knowledge bases where we have quite a different situation. This concerns, for instance, the success postulate giving top-priority to new inputs, so that all new information will have to be accepted no matter how much of the already represented information is compatible with it – something we do not want, in general, in a KS. Instead of the AGM revision operator, a non-prioritized revision operator seems to be more fundamental for a KS. Such a belief change operation is called *consolidation* in [Han91]. It proceeds in two steps: first KB is updated by the new input F, and then, if necessary, $\mathrm{KB} + F$ is consolidated, i.e. made consistent (or minimally inconsistent) by appropriate contractions.

In the sequel, we will simply say *revision* for a non-prioritized consolidating update operation, while we call the special AGM concept of revision, based on the Success postulate, *recency-preferring revision*. Thus, we will deal with three kinds of update operations differing in their method of minimizing, resp. eliminating, newly imported contradictory information:

1. Simple update, denoted by $\mathsf{Upd}(\mathrm{KB}, F)$, or in short, $\mathrm{KB} + F$, tolerating inconsistent KBs according to the principle of

   (Minimal Inconsistency) $\quad \mathrm{KB} + F$ is minimally inconsistent.

2. Consistent revision, denoted by $\mathrm{KB} \oplus F$, and described by

   (Consistency Preservation) $\quad \mathrm{KB} \oplus F$ is consistent

3. Recency-preferring revision, denoted by $\mathrm{KB} * F$, and characterized by the postulate of success, and a restricted consistency preservation postulate:

   (Success) $\quad$ If F is consistent, then $\mathrm{KB} * F \vdash F$
   (Consistency Preservation) $\quad \mathrm{KB} * F$ is only inconsistent if F is inconsistent

The AGM postulates for a revision operator $\circ = \oplus, *$, supplemented with Hansson's relevance postulate and adapted to the KS framework are:[4]

(Inclusion)	$\mathrm{KB} \circ F \leq \mathrm{KB} + F$
(Vacuity)	If F is consistent with KB, then $\mathrm{KB} \circ F \cong \mathrm{KB} + F$
(Extensionality)	$F \vdash G \;\&\; G \vdash F \;\Rightarrow\; \mathrm{KB} \circ F \cong \mathrm{KB} \circ G$
(Relevance)	$\mathrm{KB} \vdash G \;\&\; \mathrm{KB} \circ F \not\vdash G \;\Rightarrow\; (\mathrm{KB} \circ F) + G$ is inconsistent
(Conjunction 1)	$\mathrm{KB} \circ (F \wedge G) \leq (\mathrm{KB} \circ F) + G$
(Conjunction 2)	If G is consistent with $\mathrm{KB} \circ F$, then $(\mathrm{KB} \circ F) + G \leq \mathrm{KB} \circ (F \wedge G)$

8.3.1 Inconsistency in Vivid Knowledge Systems

Notice that strong negation does not satisfy the following (classically valid) condition of

(Negation as Inconsistency) $\quad \mathrm{KB} \vdash \neg F \quad$ iff $\quad \mathrm{KB} + F$ is inconsistent

[4] In [GR 92], (Inclusion) is called (Expansion 1), and (Vacuity) is called (Expansion 2).

Therefore, in contrast to classical logic, KB $\not\vdash \sim F$ does not guarantee that F is consistent with KB. Consider the extended logic program $\Pi = \{p, \sim p \leftarrow q\}$. Clearly, $\Pi \not\vdash \sim q$. However, $\Pi * q = \Pi + q$ is inconsistent. This means that several conditions for recency-preferring revision, notably (Consistency Preservation) and (Success), do not hold if it is defined according to the Levi identity,

$$\mathrm{KB} * F := (\mathrm{KB} - \sim F) + F$$

We have two options to remedy this problem. They are associated with the preferred approach to inconsistency handling: either consistency maintenance or paraconsistency, resp. minimal inconsistency. If we want consistency maintenance we could look for a definition of revision capturing the spirit of Levi's definition (something like: first make KB consistent with F, and then add F).

If, on the other hand, paraconsistency (resp. minimal inconsistency) is preferred, the Levi definition has to be modified in order to prevent the inconsistency of F in KB $* F$, e.g. in the following way:

$$\mathrm{KB} * F := \mathrm{KB} - (F \rightarrow \sim F) + F$$

For instance, if $\Pi = \{p, \sim q \leftarrow q\}$, this gives $\Pi * q = \{p, q\}$ instead of $\{p, \sim q \leftarrow q, q\}$, where the new input q is inconsistent.

8.3.2 Implicit Consolidation versus Consistency Maintenance

There is a computationally attractive alternative to global consistency maintenance: non-explosive inference operations based on mutual neutralization of contradictory information. Such inference operations[5] always yield consistent conclusions only. Thus, they can be viewed as *implicit consolidation*. While explicit consolidation is characterised by Consistency Preservation,

KB $\oplus F$ is always consistent,

implicit consolidation by means of a resp. neutralization-based inference operation $C^{\oplus}$ is characterised by the postulate of *Inherent Consistency*,

$C^{\oplus}(\mathrm{KB} + F)$ is always consistent.

The following question arises: *Given a revision operation $\oplus$, and standard inference operation C which only works for consistent KBs, is there an inherently consistent inference operation $C^{\oplus}$ such that*

$$C^{\oplus}(\mathrm{KB} + F) \quad = \quad C(\mathrm{KB} \oplus F)$$

In a KS, both $\oplus$ and $C^{\oplus}$ make sense depending on the specific requirements of an application. If a KB has to process a lot of queries, and only a few inputs, then consistency maintenance by means of a revision operation seems to be feasible. If, on the other hand, a KB has to process a lot of inputs, and only a few queries, then revision may be not feasible due to the fact that global inconsistency detection is hard to compute, and an inherently consistent inference operation might be the better choice.

[5]Investigated, e.g., in the *logic of argumentation*, and the area of *defeasible inheritance*. In [Wagner 1993a], neutralization-based top-down reasoning procedures have been proposed for extended logic programming.

Chapter 9

Further Topics, Open Problems

9.1 Planning in Active Knowledge Systems

A planning problem in an active knowledge system $A\boldsymbol{K}$ is a triple $\langle X_0, O, G\rangle$ consisting of

1. an initial state, i.e. a knowledge base $X_0 \in L_{\mathrm{KB}}$ of a KS $\boldsymbol{K}$,
2. a set of admissible operators $O \subseteq R(L^x_{\mathrm{Input}} \leftarrow L^x_{\mathrm{Query}})$, i.e. a set of action rules in $A\boldsymbol{K}$, and
3. a goal $G \in L_{\mathrm{Query}}$, being a closed query formula.

A successful plan is a sequence of admissible actions, resp. instantiated operators, $o_1, o_2, \ldots, o_n$, such that its application to the intial state yields the desired goal:

$$o_n \circ \ldots \circ o_2 \circ o_1(X_0) \vdash G$$

Obviously, STRIPS is a special case of planning in $A\boldsymbol{A}$.

Example 24 (Blocks World) $\quad X_0 = \{on(a,b),\ on(b,c),\ on(c,t),\ clear(a)\}$

$$O = \left\{ \begin{array}{l} \textbf{putonblock}(x,y): \\ on(x,y) \wedge clear(z) \wedge -on(x,z) \wedge -clear(y) \\ \leftarrow on(x,z) \wedge clear(x) \wedge clear(y) \wedge x \neq y \wedge y \neq z \wedge x \neq t \wedge y \neq t \\ \textbf{putontable}(x): \\ on(x,t) \wedge clear(z) \wedge -on(x,z) \\ \leftarrow on(x,z) \wedge clear(x) \wedge x \neq t \wedge z \neq t \end{array} \right.$$

In order to have b on a, and c not clear, we can perform the following sequence of actions:

$$puton(b,a) \circ puton(a,c) \circ putontable(b) \circ putontable(a)(X_0) \vdash on(b,a) \wedge -clear(c)$$

9.2 Active Knowledge Systems and Formal Theories of Action

Many action formalism may be mapped into active knowledge systems. We only illustrate this by means of examples.

Example 25 (Yale Shooting) *In [GL93], action domains like the following are formalized:*

$$D = \begin{cases} \textbf{initially} \sim\!\mathit{loaded},\ \mathit{alive} \\ \mathit{load}\ \textbf{causes}\ \mathit{loaded} \\ \mathit{shoot}\ \textbf{causes} \sim\!\mathit{alive}\ \textbf{if}\ \mathit{loaded} \\ \mathit{shoot}\ \textbf{causes} \sim\!\mathit{loaded} \end{cases}$$

This can be expressed with extended action rules (refering to previous events by means of PEvt*) as follows:*

$$X = \{\sim\!\mathit{loaded},\ \mathit{alive}\}$$

$$R = \begin{cases} \mathit{loaded} \leftarrow \mathsf{PEvt}(\mathit{load}) \\ \sim\!\mathit{alive} \leftarrow \mathsf{PEvt}(\mathit{shoot}), \mathit{loaded} \\ \sim\!\mathit{loaded} \leftarrow \mathsf{PEvt}(\mathit{shoot}) \end{cases}$$

9.3 Weighted Knowledge Bases

In a weighted knowledge base, an information unit is a weighted atom, resp. literal, i.e. inputs are correspondingly labelled. This allows reasoning with uncertain information. Various uncertainty formalisms could be employed for this purpose. A particular interesting one with respect to its treament of negative information seems to be the possibility theory of Dubois and Prade.

Bibliography

[Abi88] S. Abiteboul: Updates – a New Frontier, *Proc. ICDT-88*, Springer LNCS, 1988.

[AN84] A. Almukdad and D. Nelson: Constructible Falsity and Inexact Predicates, *JSL* **49**:1 (1984), 231–233.

[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.

[Bry90] F. Bry: Intensional Updates – Abduction via Deduction, *Proc. Int. Symp. on Logic Programming*, 1990.

[Cha91] S. Chakravarthy: Active Database Management Systems – Requirements, State-of-the-art, and an Evaluation, in H. Kangassalo (Ed.), *Entity-Relationship Approach: The Core of Conceptual Modelling*, Elsevier Science Publishers, 1991.

[CT92] C.B. Cross and R.H. Thomason: Conditionals and Knowledge Base Update, in P. Gärdenfors (Ed.), *Belief Revision*, Cambridge University Press, 1992, 247–275.

[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.

[FUV83] R. Fagin, J.D. Ullman and M. Vardi: On the Semantics of Updates in Databases, *Proc. ACM PODS 83*, 1983.

[Fit93] M. Fitting: Metric Methods, Three Examples and a Theorem, *J. Logic Programming*, 1993.

[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.

[GR 92] P. Gärdenfors and H. Rott: *Belief Revision*, Report 30, Konstanzer Berichte zur Logik und Wissenschaftstheorie, Universität Konstanz, 1992.

[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.

[Gel92] M. Gelfond: Logic Programming and Reasoning with Incomplete Information, *Proc. Workshop on Extensions of Logic Programming 1992*, to appear in *Annals of Mathematics and AI*.

[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.

[GL93] M. Gelfond and V. Lifschitz: Representing Actions in Extended Logic Programming, *J. Logic Programming* **17** (1993), 301–322.

[Han91] S.O. Hansson: *Belief Base Dynamics*, dissertation, Uppsala University, 1991.

[KM93] Y. Kaluzkuy and A.Y. Muravetsky: A Knowledge Representation Based on the Belnap's Four-Valued Logic, *J. of Applied Non-Classical Logics* **3**:2 (1993), 189–203.

[KM91] H. Katsuno and A.O. Mendelzon: On the Difference between Updating a Knowledge Base and Revising It, *Proc. KR'91*, Morgan Kaufmann, 1991.

[Kör66] S. Körner: *Experience and Theory*, Kegan Paul, London, 1966.

[LPS93] D. Laurent, V. Phan Luong and N. Spyratos: Updating Intensional Predicates in Deductive Databases, *Proc. 9th IEEE ICDE 93*, 1993.

[LPR93] N. Leone, L. Palopoli and M. Romeo: Updating Logic Programs, *Proc. ISMIS'93*, Springer LNAI, 1993.

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

[Lev86] H.J. Levesque: Making Believers out of Computers, *AI* **30** (1986), 81-107.

[Lev88] H.J. Levesque: Logic and the Complexitiy of Reasoning, *J. Philosophical Logic* **17** (1988), 355–389.

[LMR92] J. Lobo, J. Minker and A. Rajasekar: *Foundations of Disjunctive Logic Programming*, MIT Press, 1992.

[Min82] J. Minker: On Indefinite Databases and the Closed-World Assumption, *Proc. of CADE-82*, 1982, 292–308.

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

16–26.

[Nut88] D. Nute: Defeasible Reasoning and Decision Support Systems, *Decision Support Systems* **4** (1988), 97–110.

[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.

[PW90] D. Pearce and G. Wagner: Reasoning with Negative Information I – Strong Negation in Logic Programs, LWI Technical Report, Freie Universität Berlin, 1989. Also in L. Haaparanta, M. Kusch and I. Niiniluoto (Eds.), *Language, Knowledge, and Intentionality*, Acta Philosophica Fennica **49**, 1990.

[Pel86] F.J. Pelletier: Seventy-Five Problems for Testing Automatic Theorem Provers, *J. Automated Reasoning* **2** (1986), 191–216.

[PA92] L.M. Pereira and J.J. Alferes: Wellfounded Semantics for Logic Programs with Explicit Negation, *Proc. ECAI'92*, Wiley, 1992.

[Plo76] G.D. Plotkin: A Powerdomain Construction, *SIAM J. Comp.* **5**:3 (1976), 452–487.

[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.

[Ras94] L. Raschid: A Semantics for a Class of Stratified Production System Programs, *J. Logic Programming* (1994).

[Rei78] R. Reiter: On Closed-World Databases, in J. Minker and H. Gallaire (Eds.): *Logic and Databases*, Plenum Press, 1978.

[Rei92] R. Reiter: On Formalizing Database Updates – Preliminary Report, *Proc. EDBT-92*, 1992.

[RT88] K.A. Ross and R.W. Topor: Inferring Negative Information from Disjunctive Databases, *J. Automated Reasoning* **4**:2 (1988), 397–424.

[Sak89] C. Sakama: Possible Model Semantics for Disjunctive Databases, *Proc. 1st Int. Conf. on Deductive and Object-Oriented Databases (DOOD'89)*, North-Holland, 39–383, 1989.

[SI94] C. Sakama and K. Inoue: An Alternative Approach to the Semantics of Disjunctive Logic Programs and Deductive Databases, *J. Automated Reasoning*, forthcoming.

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

[Smy78] M. Smyth: Powerdomains, *J. Comp. Syst. Sci.* **16**:1 (1978), 23–36.

[Sti86] M. Stickel: Schubert's Steamroller, *J. Automated Reasoning* **1** (1986), 89–101.

[Thi92] E. Thijsse: *Partial Logic and Knowledge Representation*, dissertation, Tilburg University, 1992.

[Wag91a] G. Wagner: Logic Programming with Strong Negation and Inexact Predicates, *J. Logic and Computation* **1**:6 (1991), 835–859.

[Wag91b] 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.

[Wag93a] G. Wagner: Neutralization and Preemption in Extended Logic Programming, in A. Voronkov (Ed.), *Logic Programming and Automated Reasoning, Proc. LPAR'93*, Springer LNAI **698** (1993).

[Wag93b] 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: Belnap's Epistemic States and Negation-as-Failure, *Proc. of the Analyomen 2 Workshop on Negation*, Leipzig 1994, to appear with de Gruyter.

[Zan93] C. Zaniolo: On the Unification of Active Databases and Deductive Databases, in M. Worboys and A.F. Grundy (Eds.), *Advances in Databases*, Proc. of BNCOD-11, Springer LNCS **696**, 1993.