From Information Systems to Knowledge Systems[1]

G. Wagner
Institut für Informatik, Universität Leipzig
Augustusplatz 10-11, D-04109 Leipzig, Germany
E-Mail: gw@inf.fu-berlin.de, *Fax: (+49 341) 719 23 99*

Abstract

We consider the evolution of information systems towards sophisticated knowledge systems. Investigating the key concepts for information and knowledge systems from a logical point of view, we show that the two most fundamental operations are inference (query answering) and update ('data manipulation'). They have to be available in every system. Advanced information systems may allow for negative, disjunctive and uncertain information. Even more advanced systems (which might be no longer called *information*, but *knowledge systems*) will allow for sophisticated reasoning services, such as deductive query answering, active input processing, representing actions, generating explanations, diagnoses and plans.

Keywords

Knowledge systems, information growth, nonmonotonicity, deduction rules, action rules.

1 Introduction

The evolution of information system concepts can be roughly described by the sequence of *hierarchical* and *network* databases, *relational* databases, *object-oriented*, *deductive* and *active* databases, their resp. enhancements by special-purpose features, e.g. for *temporal*, *spatial*, or *uncertain*, resp. *fuzzy* information, and their 'globalization' for *distributed*, *mobile* and *cooperative* information processing. This ongoing evolution will lead to *knowledge systems* capable of processing various kinds of higher-level information, such as uncertain, disjunctive, and negative information, and in addition various kinds of knowledge, such as (deductive and active) rule knowledge, and 'social' knowledge on how to cooperate within global networks.

In order to get a clear picture and a sound foundation of these basic concepts, logical formalizations are needed. However, standard logic does not offer an adequate account

[1] in E. Falkenberg (Ed.), *Proc. IFIP Conf. on Information System Concepts (ISCO-3)*, Chapman & Hall, London 1995.

of information and knowledge processing. This is most obvious in the case of the update operation which is fundamental for information processing, and responsible for its dynamics. Standard logics are based on static semantics, and therefore not suitable for the formalization of information growth and update.

In this paper, we propose the new logical concept of a *knowledge system (KS)* extending and generalizing standard logic. Our concept of a KS accounts both for the key features of traditional information systems, such as query answering (i.e. inference) and 'data manipulation' (i.e. update), and for advanced features of future systems, such as deductive query answering, active input processing (including automated integrity and derived data maintenance), representing actions and reactions, and generating explanations, diagnoses and plans.

A KS consists essentially of two main components: an inference and an update operation manipulating knowledge bases as abstract objects.[2] In general, there are no specific restrictions on the internal structure of a knowledge base. It seems, 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. In the case of relational databases, for instance, incoming information is compiled into atomic assertions which are then stored in tabular form. We consider database systems, and any kind of information system, as (more or less low-level) knowledge systems. It turns out that practical knowledge systems are nonmonotonic, hence violating one of the basic laws of standard logic.[3] Since knowledge systems are fundamental in their own right, they are not subject to any predefined standard logic but rather define their own logic. We shall show by a number of examples that it is essential for more expressive knowledge systems to have two kinds of negation. Such systems will be called *vivid*.

The paper has the following structure: in section 2, we introduce the basic concepts, and discuss several formal properties, of knowledge systems; and in section 3, we list a number of advanced reasoning services in terms of our knowledge system concepts: deduction, action and interaction rules; explanations and diagnoses; representing actions, and planning. While section 2 gives an overview of earlier work, improves it in several points, and relates it to information systems, section 3 shows how various fields of AI can be integrated in the framework of vivid knowledge systems.

2 Knowledge Systems

2.1 Preliminaries

The language of knowledge systems consists of the logical operators conjunction ($\wedge$), disjunction ($\vee$), weak negation (alias negation-as-failure, denoted by $-$), strong negation ($\sim$), and the truth constant 1; 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.

[2] This fundamental distinction was already proposed in [Lev84] where the resp. operations were called *ASK* and *TELL*.

[3] Knowledge systems require a new kind of (generalized) model theory, such as the *nonmonotonic model theory* of [Her93, DH94].

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 well-formed formulas, respectively. 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 $\mathcal{F}$ is a set of logical operators, say $\mathcal{F} \subseteq \{1, -, \sim, \wedge, \vee, \rightarrow, \exists, \forall, \ldots\}$, then $L(\mathcal{F})$ denotes the respective set of closed well-formed formulas.

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

2.2 Basic Concepts

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

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

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

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

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

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

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

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

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

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

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

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

Also, an answer operation taking a knowledge base and an open query formula, and providing the corresponding set of valid answers, can be defined in terms of the inference relation.

Definition 2 (Answer Operation) *In* definite[6] *knowledge bases, an answer to an open query formula* $F(x_1, \ldots, x_m)$ *is a tuple, and the set of all answers is a relation:*

$$\mathsf{Ans}(\text{KB}, F(x_1, \ldots, x_m)) = \{\langle c_1, \ldots, c_m \rangle : \text{KB} \vdash F(c_1, \ldots, c_m)\}$$

where $x_1, \ldots, x_m$ *are the free variables, and* $c_1, \ldots, c_m$ *are constants.*

Not all open query formulas can be answered sensibly. We therefore require that queries are *evaluable*.[7] Answers to evaluable queries on the basis of definite KBs may be computed by means of relational algebra operations, such as projection, selection, set difference, union, and join. For instance,

$$\mathsf{Ans}(\text{KB}, F(x, y) \wedge G(y, z)) = \mathsf{Ans}(\text{KB}, F(x, y)) \bowtie \mathsf{Ans}(\text{KB}, G(y, z))$$

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

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

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

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

[6] A knowledge base $X \in L_{\text{KB}}$ is called *definite* if $X \vdash F \vee G$ implies that either $X \vdash F$, or $X \vdash G$, where $\vdash$ is a constructive inference relation (see below). If a knowledge base is indefinite, an answer to a query may be indefinite as well, corresponding to a minimal disjunctive consequence: $\mathsf{Ans}(\text{KB}, F(x)) = \text{Min}(\{T : \text{KB} \vdash \bigvee\{F(t) : t \in T\}\})$.

[7] See [vGT91] for the notion of *evaluable*, resp. *domain-independent*, formulas.

Example 1 (Relational Databases) *A relational database corresponds to a set of ground atoms. For instance,* $X_1 = \{r(S), r(P), m(P,L), m(T,S)\}$ *may represent the information that both Peter and Susan are residents, Peter is married with Linda, and Tom is married with Susan, which may also be represented by means of the following tables*

$$r = \boxed{\begin{array}{c} S \\ P \end{array}}, \quad m = \boxed{\begin{array}{cc} P & L \\ T & S \end{array}}.$$

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

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

For instance, the query "which resident is married with which non-resident ?" is answered as follows:

$$\begin{aligned} \mathsf{Ans}(X_1, m(x,y) \wedge r(x) \wedge -r(y)) &= [\mathsf{Ans}(X_1, m(x,y)) \overset{1=1}{\bowtie} \mathsf{Ans}(X_1, r(x))] \\ &\quad - [\mathsf{Ans}(X_1, m(x,y)) \overset{2=1}{\bowtie} \mathsf{Ans}(X_1, r(y))] \\ &= \{\langle P,L\rangle\} - \{\langle T,S\rangle\} = \{\langle P,L\rangle\} \end{aligned}$$

Updates are insertions, $\mathsf{Upd}(X,a) := X \cup \{a\}$, *and deletions,* $\mathsf{Upd}(X,-a) := X - \{a\}$, *of atoms. For consistent* $E \subseteq \mathrm{At} \cup \overline{\mathrm{At}}$, *we have* $\mathsf{Upd}(X,E) = X \cup E^+ - E^-$. *For example, if we learn that first Tom gets divorced from Susan, and then Peter gets divorced from Linda and marries Susan, we perform the following update:*

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

The knowledge system of relational databases, denoted by $\boldsymbol{A}$, *is then defined as*

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

It is easy to check that KS1, KS2 and KS3 hold.

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

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

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).[8] The informationally *empty KB* will be denoted by 0. By definition, $0 \leq X$ for all $X \in L_{\mathrm{KB}}$, i.e. 0 is the least element of $\langle L_{\mathrm{KB}}, \leq \rangle$.

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

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

Definition 3 (Persistent Queries) *A closed, resp. open, query formula F is called* persistent *if* $\forall X_1, X_2 \in L_{\mathrm{KB}} : X_1 \vdash F \;\Rightarrow\; X_2 \vdash F$, *resp.* $\mathsf{Ans}(X_1, F) \subseteq \mathsf{Ans}(X_2, F)$, *whenever $X_1 \leq X_2$. If all $F \in L_{\mathrm{Query}}$ are persistent, the KS and its inference relation $\vdash$ are called persistent. The set of all persistent query formulas is denoted by L_{PersQ}. An operator of the query language is called persistent, if every query formed with it and with persistent subformulas is again persistent.*

Definition 4 (Ampliative Inputs) *An input formula F is called (i)* ampliative[9] *if* $\mathrm{KB} \leq \mathsf{Upd}(\mathrm{KB}, F)$, *or (ii)* reductive *if* $\mathrm{KB} \geq \mathsf{Upd}(\mathrm{KB}, F)$. *A KS and its update operation* Upd *are called* ampliative, *if all inputs $F \in L_{\mathrm{Input}}$ are ampliative. The set of all ampliative input formulas is denoted by L_{AmpI}.*

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

Definition 5 (Regular Knowledge System) *A knowledge system $\boldsymbol{K}$ is called* regular, *if there is a preorder $\langle L_{\mathrm{KB}}, \leq, 0 \rangle$ with least element 0, a designated set $L_{\mathrm{Unit}} \subseteq L_{\mathrm{Input}}$, and a function $\mathrm{Inc} : L_{\mathrm{KB}} \to 2^{L_{\mathrm{Unit}}}$, such that*

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

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

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

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

(KS7) For any $X \in L_{\mathrm{KB}}$, and $F \in L_{\mathrm{Input}}$, $X \vdash F$ implies $\mathrm{Inc}(X + F) \subseteq \mathrm{Inc}(X)$.

A regular KS will be represented as a 9-tuple

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

Obviously, the knowledge system $\boldsymbol{A}$ of relational databases is regular: the information ordering is given by set inclusion, the informationally empty database is the empty set, the information units are atoms, and the inconsistency function is constantly empty, since contradictions cannot occur in a relational database where only positive information is stored (the possible violation of integrity constraints is a different problem).

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

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*

$$X_2 = \{r(S), r(P), s(S), \sim s(L), \sim s(P),\ m(P,L),\ m(T,S)\}$$

which is also represented by the following tables:

$$r = \boxed{\begin{array}{c} S \\ P \end{array}},\ s = \boxed{S},\ \tilde{s} = \boxed{\begin{array}{c} L \\ P \end{array}},\ m = \boxed{\begin{array}{cc} P & L \\ T & S \end{array}},$$

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

Notice that for certain predicates, such as nonsmoker, *negative information is explicitly stored in the fact base, i.e. a fact base may represent 'incomplete' predicates. The CWA,*[10] *from $-p(c)$ we may conclude that $\sim p(c)$, does no longer hold globally for all predicates. While we can certainly not assume the CWA for empirical predicates like* smoker, *we should also not assume it for the relation* married *in some local KB since people may get married all over the world, and thus* married *will not be totally represented in a local KB. We may assume the CWA, however, for a predicate like* resident, *simply because all residents of a city are registered in the local KB of that city.*

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 is defined in the following way:[11]

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

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

$$\mathsf{Ans}(X_2,\ m(x,y) \wedge \sim s(y) \wedge -r(y)) = \mathsf{Ans}(X_2, m(x,y)) \stackrel{2=1}{\bowtie} [\mathsf{Ans}(X_2, \tilde{s}(y)) - \mathsf{Ans}(X_2, r(y))]$$

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

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

---

[10] The Closed-World Assumption, in a specific form for relational databases, was originally proposed in [Rei78]. In [Wag94a], we propose a more general form of the CWA which 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.

[11] This inductive definition is completed by DeMorgan-style rewrite rules including double negation rules such as $\sim -a \longrightarrow a$. See [Wag94b].

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

Whenever we deal with both kinds of negation, the strong negation $\sim$ is the principal negation operator (expressing explicit falsity), and the weak negation $-$ is an auxiliary negation operator used, e.g., to express the CWA.

2.3 Basic Properties of Inference and Update

Definition 6 (Constructive Inference) *Let $L \subseteq \mathrm{Lit}$ be an arbitrary set of 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: for any $F, G \in L_{\mathrm{Query}}$: (i) $\lceil L \rceil \vdash F \vee G$ implies $\lceil L \rceil \vdash F$ or $\lceil L \rceil \vdash G$; (ii) $\lceil L \rceil \vdash \sim (F \wedge G)$ implies $\lceil L \rceil \vdash \sim F$ or $\lceil L \rceil \vdash \sim G$.*

The property of constructive inference guarantees that, on the basis of definite knowledge, query formulas are decomposable. Obviously, the first condition (of constructible truth) excludes the possibility of certain disjunctive tautologies such as the classical *tertium non datur*, whereas its negative counterpart excludes, for instance, the dual *principle of contradiction*.

The next property (due to Urbas [Urb90]) excludes the possibility of trivial inferences, i.e. non-tautological inferences which are solely based on the form of a KB and a query and not on their content. For example, $\{s(L), \sim s(L)\} \vdash m(P, S)$ is such a trivial inference which is valid in classical logic, i.e. from contradictory information on Linda being a smoker, we may infer that Peter is married with Susan, and thus we would get (infinitely many) unsensible answers to any query. This is clearly undesirable in a knowledge system.

Definition 7 (Tautology) *$F \in L_{\mathrm{Query}}$ is called a* tautology *in a knowledge system, if $X \vdash F$ for all $X \in L_{\mathrm{KB}}$.*

Definition 8 (Non-Explosive Inference) *An inference relation $\vdash$ is called* non-explosive *if for every non-tautology $F \in L_{\mathrm{Query}}$, and for every knowledge base $X > 0$, there is a variant F' of F (obtained by uniform substitution of propositional constituents) such that $X \not\vdash F'$.*

While most standard logics allow for trivial inferences, their positive fragments and certain paraconsistent logics, such as Belnap's [Bel77] four-valued, or Nelson's [AN84] paraconsistent constructive logic, are non-explosive.

The following important property guarantees the freedom of knowledge base evolution.

Definition 9 (Input Completeness) *A KS is called* input complete *if*

$$\forall X_1, X_2 \in L_{\mathrm{KB}} \exists F \in L_{\mathrm{Input}} : X_2 = \mathsf{Upd}(X_1, F)$$

Observation 1 *A KS is input complete iff KBs are both input constructible and input destructible, i.e. both of the following conditions hold:*

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

Further formal properties of knowledge systems are discussed in [Wag94b]. It is easy to check that $\boldsymbol{F}$, the system of fact bases, satisfies Constructive and Non-Explosive Inference, and is input complete. For more complex cases of knowledge systems, including disjunctive, deductive and active knowledge systems, see [Wag94b].

2.4 Nonmonotonicity

The following definition 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.

Definition 10 (Monotonicity) *A KS is called* monotonic *if for all $X \in L_{\mathrm{KB}}$, and all $F \in L_{\mathrm{Input}}$, we have $C(X) \subseteq C(\mathsf{Upd}(X, F))$.*

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

There are two 'parameters' on which Monotonicity depends: the update operation may be ampliative or not, and the inference relation may be persistent or not.

Observation 2 *A KS is monotonic if it is ampliative and persistent.*[12]

Practical systems will be nonmonotonic since they will allow for non-persistent queries (e.g. by means of negation-as-failure), and for non-ampliative updates (e.g. by means of deletion/contraction).

3 Advanced Reasoning Services

As opposed to database systems where the only reasoning service is query answering, more advanced knowledge systems offer in addition a number of advanced reasoning services such as deductive query answering by means of deduction rules, active input processing by means of action rules, cooperative services by means of interaction rules, generating explanations, diagnoses and plans (based on the ability to represent actions). We briefly sketch these KS services whose full functionality will be only available in vivid knowledge systems because they (more or less) depend on the availability of both weak negation (for expressing deletion/contraction and the CWA) and strong negation (for expressing explicit falsity and inconsistency).

[12] Or, rather exotically, if all inputs are reductive and all queries are 'antipersistent', i.e. preserved under information decrease. It is still an open problem, whether – or under which conditions – the converse holds.

3.1 Knowledge Services Based on Rules

We propose four kinds of rules:

Deduction Rules can be used to define intensional relations in terms of base relations, express subsumption relationships between concepts, or causal relationships between causes and effects.

Action Rules can be used to represent actions for the purpose of planning, or to express integrity, and derived data, maintenance policies.

Reaction Rules can be used to specify the reactive behavior of a knowledge base (resp. knowledge-based agent) when it receives messages, either from the external world, or from other nodes of a network it participates in.

Interaction Rules can be used to specify the communicative behavior of a knowledge base (resp. knowledge-based agent).

By implying a genuine concept of rules, knowledge systems confirm the *facts and rules paradigm* of logic programming. Unlike rather naive approaches, where a rule is interpreted as a material implication formula of some standard logic, the concept of knowledge systems suggests to regard a rule $r = F \leftarrow G$ as a specific update function operating on knowledge bases:[13]

$$r(\mathrm{KB}) = \begin{cases} \mathsf{Upd}(\mathrm{KB}, F) & \text{if } \mathrm{KB} \vdash G \\ \mathrm{KB} & \text{otherwise} \end{cases}$$

where $F \in L_{\mathrm{Input}}$ and $G \in L_{\mathrm{Query}}$. For non-ground rules $r = F(x) \leftarrow G(x)$, application is defined by

$$r(\mathrm{KB}) := \mathsf{Upd}(\mathrm{KB}, \{F(c) : c \in Dom(\mathrm{KB}) \;\&\; \mathrm{KB} \vdash G(c))\})$$

where it is required that $\mathrm{Free}(F) \subseteq \mathrm{Free}(G)$, $F \in L^x_{\mathrm{Input}}$, and $G \in L^x_{\mathrm{Query}}$ is evaluable (such rules are called *range-restricted*).

Definition 11 (Rule Satisfaction) *We say that a knowledge base satisfies a rule if it is a fixpoint of it. Formally,* KB **satisfies** r *iff* $r(\mathrm{KB}) = \mathrm{KB}$. KB *is* **closed** *under a set of rules R, if it is a common fixpoint of all rules:* $r(\mathrm{KB}) = \mathrm{KB}$*, for all* $r \in R$.

In general, closures defined in this way may contain too much, and therefore one prefers certain appropriately computed closures over others. As a general preference criterion for rule-based closures, we propose the *stability* of rule application. This leads to the *stable closure semantics* which was shown in [Wag94b] to be a generalization of the stable model semantics for normal logic programs of [GL88]. In the sequel, we shall assume that there is a unique stable closure of a knowledge base X under a set of rules R, and denote it by $R(X)$ (in certain – rather unusual – cases, there might be several stable closures, or even none).

[13]This follows Belnap's idea of an 'information state', see [Bel77].

3.1.1 Deductive Knowledge Bases

A *deductive knowledge base* is a pair $\langle X, R\rangle$ consisting of a knowledge base $X \in L_{\mathrm{KB}}$ and a set R of range-restricted rules, called *deduction rules*. In the basic setting, deduction rules are not affected by updates (i.e. only extensional predicates may be updated):

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

But deduction rules help to answer queries:

$$\langle X, R\rangle \vdash F \quad \text{iff} \quad R(X) \vdash F$$

Example 3 *Let the rule set* $R_1 = \{m(y, x) \leftarrow m(x, y)\}$ *contain a single rule expressing the symmetry of the* married *relation, and* X_1 *be the relational database of example 1. Then,* $\langle X_1, R_1\rangle$ *is a monotonic deductive database since there is no rule with negation-as-failure. Therefore, we obtain a least deductive closure (being the unique stable closure):*

$$R_1(X_1) = X_1 \cup \{m(L, P),\ m(S, T)\}$$

3.1.2 Active Knowledge Bases

An *active knowledge base* is a pair $\langle X, R\rangle$ consisting of a knowledge base $X \in L_{\mathrm{KB}}$ and a set R of range-restricted rules, called *action rules*. In any legal state, an active knowledge base is closed: $X = R(X)$. Action rules do not participate in query answering:

$$\langle X, R\rangle \vdash F \quad \text{iff} \quad X \vdash F$$

But action rules help to process inputs by actively closing the updated KB:

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

thus allowing, e.g., for automatic integrity, and derived data, maintenance (which is achieved by 'triggers' in SQL databases).

Example 4 *Let* $X_4 = \{on(a, t),\ clear(a)\}$ *describe the situation where block* a *is on the table, and is clear, and* $r_4 = -clear(x) \leftarrow on(y, x)\}$ *express the integrity maintenance rule that block* x *is no longer clear, as soon as some block* y *has been put on it. Then,* $\langle X_4, R_4\rangle$*, where* $R_4 = \{r_4\}$*, denotes the corresponding active database. If a robot puts a new block, say* b*, on block* a*, this leads to the update* $\mathsf{Upd}(\langle X_4, R_4\rangle,\ on(b, a))$*, triggering the integrity maintenance action rule* r_4 *in the following way:*

$$\begin{aligned}
\mathsf{Upd}(\langle X_4, R_4\rangle,\ on(b, a)) &= \langle R_4(X_4 \cup \{on(b, a)\}), R_4\rangle \\
&= \langle r_4(\{on(a, t),\ clear(a),\ on(b, a)\}),\ R_4\rangle \\
&= \langle \{on(a, t),\ on(b, a)\},\ R_4\rangle
\end{aligned}$$

3.1.3 Reaction and Interaction Rules

Rules can also be used to specify the reactive and interactive behavior of a KB when it receives messages, either from the external world, or from other nodes of a network it participates in. This allows to specify cooperative knowledge bases in a network of active KBs, or, more generally, knowledge-based agents in a multi-agent society.

Reaction rules control the behaviour of a KB in response to perception and communication events. A reaction rule has the form

$$Action \ \leftarrow \ \mathsf{recvMsg}(Msg,\, Sender),\ Query$$

where *Action* is any combination of an update, private action, or communication act, $Query \in L^x_{\text{Query}}$ is the epistemic condition of the reaction, and $\mathsf{recvMsg}(Msg,\, Sender)$ is a test whether the event buffer contains the incoming message *Msg* addressed to the AKB by *Sender*.

An *interaction rule* is a reaction rule such that its conclusion represents a communication act. Interaction rules have the form:

$$\mathsf{sendMsg}(OutMsg,\, Receiver),\ Input \ \leftarrow \ \mathsf{recvMsg}(InMsg,\, Sender),\ Query$$

where $Input \in L^x_{\text{Input}}$ represents the epistemic effect of the interaction, $Query \in L^x_{\text{Query}}$ is its epistemic condition, $\mathsf{recvMsg}(InMsg,\, Sender)$ is a condition on the communication events triggering the interaction rule, and $\mathsf{sendMsg}(\ldots)$ represents a communication act which is realized by asynchronous message passing.

In SQL databases, some (rather limited) form of interaction, resp. cooperation, can be achieved by means of 'remote procedure calls'.

3.2 Explanations

A set of formulas $E \subseteq L_{\text{Input}}$ is an *explanation* of a set of observations $O \subseteq L_{\text{Query}}$ with respect to $X \in L_{\text{KB}}$, symbolically $E \in \mathsf{Expl}(X, O)$, if

(1) E yields O on the basis of X: $X + E \vdash \bigwedge O$, and
(2) E is relevant for obtaining O: $X - E \not\vdash \bigwedge O$.

Notice that (a) since being more realistic and more general than other approaches, we do neither require that X, nor that E is consistent; and (b) if $O \subseteq L_{\text{Input}}$, observations trivially explain themselves: $O \in \mathsf{Expl}(X, O)$.

Which explanations are to be preferred ? A possible preference criterion is *minimal specificity:* E is considered more specific than F (wrt X) if E is an explanation of F in X, i.e. if $E \in \mathsf{Expl}(X, F)$.

There might be a 'bias', i.e. a set $B \subseteq L_{\text{Input}}$ of possible hypothesis from which the explanation has to come. Explanations with respect to a bias B are defined as $\mathsf{Expl}(X, O, B) := \mathsf{Expl}(X, O) \cap 2^B$.

Example 5 (Knowledge Discovery) *The process of finding an explanation (in terms of certain predicates) of a specific subset of a given database is an example of* knowledge discovery in databases (KDD). *It can be achieved, for instance, by applying techniques* *from* inductive logic programming *where from a set of positive and negative examples* $O \subseteq$ XLit, *and a 'background theory' (say, a deductive fact base* $\langle X, R\rangle$*), a set of rules* E *which would explain the examples:* $\langle X, R \cup E\rangle \vdash \bigwedge O$, *is induced. Starting, e.g., with a deductive fact base describing that Tweety1 and Tweety2 are birds, Opus is a penguin, Tweety1 does fly, Tweety2 does not fly, penguins are birds, and penguins do not fly:*

$$\begin{array}{rcl} X & = & \{b(T_1), b(T_2), p(O), f(T_1), \sim f(T_2)\} \\ R & = & \{b(x) \leftarrow p(x),\ \sim f(x) \leftarrow p(x)\} \end{array}$$

we could choose $O = \{f(T_1), \sim f(T_2)\}$ *as the set of facts to be explained on the basis of the background theory* $\langle X - O, R\rangle$. *An explanation for* O *would be*

$$E = \{p(T_2),\ f(x) \leftarrow b(x) \wedge -\!\sim f(x)\}$$

3.3 Diagnoses

The concept of a *consistency-based diagnoses* has usually been defined with respect to a *system description* $X \in L_{\mathrm{KB}}$, including the specification of a set of system components by means of the predicate $component(x)$. In a normal mode of operation, all system components work properly which can be expressed by the following default deduction rule:

$$r_{ok} \ = \ ok(x) \leftarrow component(x) \wedge -\!\sim ok(x)$$

Normally, $\langle X, \{r_{ok}\}\rangle$ is consistent with all observations concerning the system behavior. It is the *abnormal* mode of operation where we make observations $O \subseteq L_{\mathrm{Input}}$, contrary to our expectations and not consistent with the assumption that all components are ok, i.e. O is inconsistent with $\langle X, \{r_{ok}\}\rangle$.

A diagnosis is defined as a set of non-ok-statements specifying the set of components assumed to be faulty, and responsible for the inconsistency,

$$D = \{\sim ok(c_1), \ldots, \sim ok(c_n)\}$$

such that $X \vdash component(c_i)$ for $i = 1, \ldots, n$, and O is consistent with $\langle X + D, \{r_{ok}\}\rangle$, formally: $\mathrm{Inc}(r_{ok}(X + D + O)) \subseteq \mathrm{Inc}(r_{ok}(X + D))$. We then write $D \in \mathsf{Diag}(X, O)$.

Of course, an explanation for O in terms of $\sim ok$-literals should also count as a diagnosis, and indeed, we have the following

Observation 3 $\mathsf{Expl}(X, O, \{\sim ok(x) : X \vdash component(x)\}) \subseteq \mathsf{Diag}(X, O)$.

Proof sketch: Let $D \in \mathsf{Expl}(X, O, \{\sim ok(x) : X \vdash component(x)\})$, then $X + D \vdash \bigwedge O$, implying that $\mathrm{Inc}(X + D + O) \subseteq \mathrm{Inc}(X + D)$, by (KS7). $\Box$

3.4 Representing Actions and Generating Plans

In automated planning, a plan is constructed as a sequence of actions such that when executed in an initial situation, it changes the situation in such a way that after the execution a certain pre-specified goal holds. This requires

1) the ability to represent, and reason on the basis of, situations, and

2) the ability to represent actions epistemically, and to simulate their execution (on the epistemic level) in order to test their effects on certain situations.

The usual logical representation of an action α consists of a precondition P_α which has to be satisfied in order to perform α successfully, and a postcondition C_α which is guaranteed to hold after α has been performed successfully. In other words, P_α represents the epistemic condition, and C_α represents the epistemic effects, of α. Thus, an action can be represented by an action rule: $r_\alpha : C_\alpha \leftarrow P_\alpha$. For instance, the action rule

$$\begin{array}{l}\mathsf{putontable}(x) : \\ on(x,t) \wedge clear(z) \wedge -on(x,z) \;\leftarrow\; on(x,z) \wedge clear(x) \wedge x \neq t \wedge z \neq t\end{array}$$

corresponds to the 'blocks world' action *put block x on the table.*

A **planning system** consists of a knowledge system $\boldsymbol{K}$, a set L_{Act} of all available action types $\alpha_1, \alpha_2, \ldots$, and a corresponding set of action rules $R = \{r_{\alpha_1}, r_{\alpha_2}, \ldots\}$.

A planning problem on the basis of a planning system is defined as a pair $\langle X_0, G\rangle$ consisting of

1) a knowledge base $X_0 \in L_{\mathrm{KB}}$, describing the initial situation,

2) a goal $G \in L_{\mathrm{Query}}$.

A *linear plan* π can then be defined as being composed of a sequence of elementary actions (i.e. instantiated action rules)

$$\pi = \alpha_n(t_n) \circ \ldots \circ \alpha_2(t_2) \circ \alpha_1(t_1)$$

such that its application (as a composed function) to the initial knowledge state X_0 yields the desired goal G:

$$\pi(X_0) \vdash G$$

Example 6 (Blocks World) *If we have in addition to the operator* putontable(x), *described above, a second operator*

$$\begin{array}{l}\mathrm{putonblock}(x,y) : \\ on(x,y) \wedge clear(z) \wedge -on(x,z) \wedge -clear(y) \\ \leftarrow on(x,z) \wedge clear(x) \wedge clear(y) \wedge x \neq y \wedge y \neq z \wedge x \neq t \wedge y \neq t\end{array}$$

we can pursue the goal G that a is on b, b is on c, and c is on the table,

$$G = on(a,b) \wedge on(b,c) \wedge on(c,t)$$

on the basis of the initial knowledge base

$$X_0 = \{on(c,a),\ on(a,t),\ on(b,t),\ clear(b),\ clear(c)\}$$

by the following plan:

$$\pi_{abc} \;=\; \mathrm{putonblock}(c,a) \circ \mathrm{putonblock}(a,b) \circ \mathrm{putontable}(c)$$

whose application proves to be successful:

$$\pi_{abc}(X_0) = \{on(c,t),\ on(b,c),\ on(a,b),\ clear(a)\}$$

and consequently,

$$\pi_{abc}(X_0) \vdash G$$

4 Conclusion

We have shown that the requirements of information and knowledge processing lead to a new logical framework deviating from standard logics in several points. Lacking the necessary space, we have avoided to discuss the connections to other logical systems – notably partial logics – in this paper. We would like to stress that knowledge-based systems (and many AI systems like diagnosis, planning, or multi-agent systems) have to evolve from (resp. integrated with) information system technology.

Our concept of a knowledge system subsumes existing database and information systems on the logical level only. In order to realize practical knowledge base management systems corresponding to our knowledge system concepts many additional questions will arise. But all these questions can only be addressed properly after the key concepts for knowledge systems have been clarified and formalized logically, and we hope that our proposal will be recognized as a contribution to this.

Acknowledgements
Parts of this work have been done during a research stay at the Institut de Recherche en Informatique de Toulouse (IRIT) in 1994. I have also benefited from discussions on knowledge systems with Heinrich Herre. Finally, I want to thank one of the anonymous referees for his helpful comments and suggestions.

References

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

[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.

[DH94] J. Dietrich and H. Herre: Outline of Nonmonotonic Model Theory, Preprint Report 5/1994, Universität Leipzig, Naturwissenchaftlich-Theoretisches Zentrum.

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

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

[Her93] H. Herre: Contributions to Nonmonotonic Model Theory, in V. Marek, A. Nerode and P.H. Schmitt (Eds.), *Proc. Workshop on Nonclassical Logics in Computer Science*, Dagstuhl Bericht, 1993.

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

[Lev86] H.J. Levesque: Making Believers out of Computers, *Artificial Intelligence* **30** (1986), 81-107.

[Rei78] R. Reiter: On Closed-World Databases, in J. Minker and H. Gallaire (Eds.): *Logic and Databases*, Plenum Press, 1978.

[Urb90] I. Urbas: Paraconsistency, *Studies in Soviet Thought* **39** (1990), Kluwer, 343–354.

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

[Wag94b] G. Wagner: *Disjunctive, Deductive and Active Knowledge Bases*, LWI Report 22/1994, Freie Universität Berlin; available as `ddakbs.dvi.gz`, resp. `ddakbs.ps.gz`, via ftp from `ftp.inf.fu-berlin.de` in the directory `/pub/reports/wagner` (notice that the file has to be 'gunzip'ed).

Dr. Wagner is currently an assistant professor at the computer science department of the university of Leipzig. His research interests comprise the logical and conceptual foundations of information and knowledge systems, nonmonotonic reasoning, and multiagent systems. After finishing his PhD at the Free University of Berlin in 1993, he was a guest researcher at the Institut de Recherche en Informatique de Toulouse (IRIT) in 1994, and a HCM post-doc fellow at the New University of Lisbon in 1995.