How an Agent Might Think

Andrzej Szałas, *Institute of Informatics, University of Warsaw, 02-097 Warsaw, Banacha 2, Poland, and Department of Computer and Information Science, Linköping University, S-581 83 Linköping, Sweden, email: andrzej.szalas@{mimuw.edu.pl, liu.se}*

Abstract

The current paper is devoted to extensions of the rule query language 4QL proposed by Małuszyński and Szałas. 4QL is a DATALOG$^{\neg\neg}$-like language, allowing one to use rules with negation in heads and bodies of rules. It is based on a simple and intuitive semantics and provides uniform tools for lightweight versions of well-known forms of nonmonotonic reasoning. In addition, 4QL is tractable w.r.t. data complexity and captures PTIME queries.

In the current paper we relax most of restrictions of 4QL, obtaining a powerful but still tractable query language 4QL^{+}. In its development we mainly focused on its pragmatic aspects: simplicity, tractability and generality. In the paper we discuss our approach and choices made, define a new, more general semantics and investigate properties of 4QL^{+}.

Keywords: Rule language, DATALOG, DATALOG$^{\neg\neg}$, Nonmonotonic reasoning, Paraconsistent semantics, Agent systems

1 Contents, Contribution and Structure of the Paper

What does it mean that an (artificially) intelligent agent thinks? We do not intend here to answer such a general (and not yet really answered) question, focusing on its rather narrow sense. That is, for the purpose of our paper "thinking" is the process of modeling the world by establishing related facts and theories; and reasoning about the world by deriving conclusions on the basis of constructed models. There are then three major aspects we focus on: how facts are collected and reported, how are theories established and how conclusions are derived. From the perspective of the current paper facts are being told, e.g., by agents through speech acts or by the environment through perception. We assume that theories modeling the world are provided by developers of agent systems. Our concern is to define a framework rich enough to represent complex knowledge bases,[1] equipped with tractable reasoning machinery and capturing all tractable reasoning schemata. Some fundamental steps towards these directions have already been made and reported in [27, 28], where a query language, called 4QL, has been defined and investigated. In the current paper we generalize ideas of [27, 28].

The environments in which agents' knowledge bases are used frequently consist of multiple information sources, so one has to address problems of missing and/or inconsistent knowledge. Technologies adequately dealing with such circumstances should provide tools for completing gaps in knowledge and for disambiguation of inconsistencies. Typically, one uses forms of nonmonotonic/defeasible reasoning techniques to serve these purposes [13, 26, 30, 33]. When performance of real world intelligent systems is concerned, pragmatic uses of logics require efficient modeling and reasoning. Therefore tractability is one of our prime prerequisites. It can hardly be achieved even in classical propositional logic, without placing cer-

[1] We use the term "knowledge base" as in the area of knowledge representation rather than in theories about knowledge and belief.

tain restrictions on formulas used in knowledge bases. The situation is even worse, when nonmonotonic or multimodal formalisms are used in their full generality [3, 14, 20, 22, 24, 34]. Already in 4QL, extended in this paper, such forms of reasoning became tractable. This is achieved by restricting formulas to rules and introducing modules and external literals. In the new language, 4QL$^+$, we considerably relax restrictions of 4QL, still remaining within a tractable framework.

Note that inconsistencies require a form of paraconsistent reasoning. Let us emphasize that paraconsistency has not been a goal of this research, being rather its unavoidable consequence. However, we deal with quite simple and natural paraconsistent semantics. For discussions of more advanced techniques, see [5, 6, 9, 10, 15].

In the current paper we mainly focus on agent systems. In such cases modeling beliefs, perception and interaction patterns is indeed a complex task [20, 22, 25, 31]. We believe that the proposed language adds a value to these approaches by a uniform treatment of monotonic and nonmonotonic reasoning techniques and founding the reasoning on knowledge based systems. To our knowledge there is no declarative approach to agent systems enjoying the properties of 4QL$^+$ (for alternative approaches see, e.g., [2, 12] and references there).

The original contribution of the paper depends on extending 4QL by:

- multisource queries to heterogeneous databases;
- allowing rules with bodies expressed by arbitrary (multisource) first-order formulas;
- substantially relaxing the requirements concerning the layered architecture.

These syntactic changes are accompanied by new definitions of semantics. We have also provided a new PTIME algorithm for computing well-supported models and working with such syntactic extensions. Let us emphasize that the proposed semantics is rather general as, in its substantial part, it abstracts from 4QL$^+$ itself. This allows us to incorporate heterogeneous information sources not necessarily defined by means of 4QL$^+$. The only requirement is that information sources answer queries by providing truth values or sets of ground literals, where such sets are finite but unrestricted in any other sense. In particular, they may contain contradictory literals or no literals expressing a particular fact or its negation.

The paper is structured as follows. First, in Section 2, we discuss the approach of [7, 8] and motivate our choices. Then, in Section 3, we provide a discussion of an alternative approach, fundamental to the current paper. In Section 4 we remind the 4QL query language defined and investigated in [27, 28]. Section 5 is devoted to discussion of extensions of 4QL. In Section 6, the 4QL+ query language is defined and investigated. Finally, Section 7 concludes the paper.

2 What is an Agent Being Told

In [7, 8] a four-valued logic is proposed as a formalization of reasoning principles when a computer or an agent receives information from multiple sources. The reasoning is founded on lattices A_4 and L_4 shown in Figure 1, where $\mathbf{t}, \mathbf{b}, \mathbf{n}, \mathbf{f}$ stand for *true, both, none* and *false*, respectively. The role of L_4 is that it provides basis for reasoning, while A_4 reflects the process of gathering information from various sources.

In this paper we will need first-order language without function symbols (however, allowing constants). That is, we allow *atomic formulas* of the form $r(\bar{u})$, where r is a relation symbol and $\bar{u}$ is a tuple consisting of variables and/or constants. Complex formulas are built using propositional connectives and first order quantifiers $\forall, \exists$. To define the semantics we

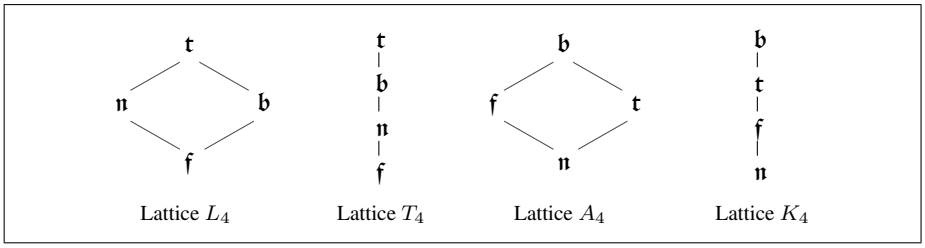

FIG. 1. Lattices discussed in the paper.

need some definitions. By a *literal* we understand an atomic formula or the negation of an atomic formula. A *ground literal* is a literal without variables. For notational convenience we identify $\neg\neg\ell$ with its equivalent ℓ.

The following definition explains the terminology of truth values.

DEFINITION 2.1
By an *interpretation* we mean any set of ground literals. The *truth value* of a literal ℓ in interpretation $\mathcal{I}$, denoted by $\mathcal{I}(\ell)$, is the value defined as follows:

$$\mathcal{I}(\ell) \stackrel{\text{def}}{=} \begin{cases} \mathfrak{t} & \text{if } \ell \in \mathcal{I} \text{ and } (\neg\ell) \notin \mathcal{I}; \\ \mathfrak{b} & \text{if } \ell \in \mathcal{I} \text{ and } (\neg\ell) \in \mathcal{I}; \\ \mathfrak{n} & \text{if } \ell \notin \mathcal{I} \text{ and } (\neg\ell) \notin \mathcal{I}; \\ \mathfrak{f} & \text{if } \ell \notin \mathcal{I} \text{ and } (\neg\ell) \in \mathcal{I}. \end{cases}$$ ◁

There has been some misunderstanding behind the intuitive meaning of reasoning based on L_4. For discussions and explanations see, e.g., [7, 8, 17, 18, 23, 38, 39]. As indicated in [4], the relevance logic FDE (First-Degree Entailment) is semantically determined by L_4, assuming that connectives $\neg, \wedge_{L_4}, \vee_{L_4}, \rightarrow_{L_4}$ are defined by:

$$\begin{array}{l} \neg\mathfrak{t} = \mathfrak{f}, \;\; \neg\mathfrak{b} = \mathfrak{b}, \;\; \neg\mathfrak{n} = \mathfrak{n}, \;\; \neg\mathfrak{f} = \mathfrak{t}, \\ t \wedge_{L_4} t' \stackrel{\text{def}}{=} \min_{L_4}\{t, t'\}, \;\; t \vee_{L_4} t' \stackrel{\text{def}}{=} \max_{L_4}\{t, t'\}, \;\; t \rightarrow_{L_4} t' \stackrel{\text{def}}{\equiv} t \leq_{L_4} t', \end{array} \tag{2.1}$$

where $t, t' \in \{\mathfrak{t}, \mathfrak{b}, \mathfrak{n}, \mathfrak{f}\}$, $\leq_{L_4}$ is the order of lattice L_4 and $\min_{L_4}$ ($\max_{L_4}$) stands for the greatest lower bound (the least upper bound, respectively) w.r.t. $\leq_{L_4}$. As pointed out in [39],

> there is only one canonical and non-metaphorical account of the four values $\mathfrak{t}, \mathfrak{b}, \mathfrak{n}, \mathfrak{f}$: they are *told values*, representing what *as a matter of fact* the computer has been told.

Now observe that $\mathfrak{b} \vee_{L_4} \mathfrak{n} = \mathfrak{t}$. The intuition behind this somehow odd result is the following [23]:

> if $\mathcal{I}(A) = \mathfrak{b}$ then we have (in particular) been told that A is true, and so that $(A \vee_{L_4} B)$ is true; and if $\mathcal{I}(B) = \mathfrak{n}$ then we have not been told that B is $\mathfrak{f}$, nor, therefore, that $(A \vee_{L_4} B)$ is $\mathfrak{f}$; so $(A \vee_{L_4} B)$ is true. (2.2)

Let us have a closer look at the above reasoning. It applies to the situation when one information source told us A and another told us $\neg A$. However, when a source tells us both A and $\neg A$, i.e., that the truth value of A is $\mathfrak{b}$, we cannot assume that *in particular* we have been told

that A is true. The explanation (2.2) does not address this case.[2] In fact, the following example, based on one provided in [32], shows that such results can be misleading, as deviating too far from the usual understanding of connectives and pragmatics of their use.

EXAMPLE 2.2
Consider a web service supplying information about stocks. Assume that a web agent looks for stock portfolios of low risk or promising big gain. The agent's query can be expressed by $(lr(X) \vee_{L_4} bg(X))$, where lr and bg stand for "low risk" and "big gain", respectively. For simplicity, assume that the service has been told the following facts:

$$lr(s_1), lr(s_2), \neg lr(s_2).$$

The query looks for stock portfolios X that are instances of $(lr(X) \vee_{L_4} bg(X))$ w.r.t. the knowledge base. This disjunction is true for $X = s_1$ as well as for $X = s_2$.

The answer $X = s_1$ for the query is intuitively correct. On the other hand, one also receives an answer that the required disjunction is true for s_2 solely on the basis of two contradictory facts $lr(s_2), \neg lr(s_2)$ which is not really what one would expect. ◁

The above example shows that definition $\mathfrak{b} \vee_{L_4} \mathfrak{n} = \mathfrak{t}$ does not reflect intuitions in application areas we focus on. Similar objections concern $\mathfrak{b} \wedge_{L_4} \mathfrak{n} = \mathfrak{f}$.

There are, of course, situations where contradictions (expressed by $\mathfrak{b}$) can easily be resolved, e.g., by examining information sources. In particular, a source may be more reliable than another one or may have more authority than another; more specific rules are typically preferred, etc. Also, the lack of information (expressed by $\mathfrak{n}$) can be resolved, e.g., by using defaults, autoepistemic reasoning, etc. However, neither FDE nor its variants provide formal means for disambiguation of contradictions or for completing missing information. Such means inevitably lead to nonmonotonicity of reasoning.

3 An Alternative to L_4

As discussed in Section 2, in many application areas L_4 is not adequate. We claim here that the pragmatics of disjunction should include the following principles:

$$\begin{array}{l} \vdash \text{disjunction is true only when at least one of its operands is true;} \\ \vdash \text{disjunction is false only when all its operands are false;} \end{array} \tag{3.1}$$

and the pragmatics of conjunction:

$$\begin{array}{l} \vdash \text{conjunction is true only when all its operands are true;} \\ \vdash \text{conjunction is false only when at least one of its operands is false.} \end{array} \tag{3.2}$$

Apparently, disjunction and conjunction, based on L_4 and defined by (2.1), do not conform to these principles. Therefore the following ordering on $\mathfrak{t}, \mathfrak{b}, \mathfrak{n}, \mathfrak{f}$ is considered in [16, 27, 28, 29] as a suitable alternative:

$$\mathfrak{f} < \mathfrak{n} < \mathfrak{b} < \mathfrak{t}. \tag{3.3}$$

We further denote this ordering by T_4 (see also Figure 1).

We assume that $\mathfrak{t}$ and $\mathfrak{b}$ are designated values.[3] This distinguishes the current approach from [27, 28, 29], where $\mathfrak{t}$ is the only designated value. Note that the change of designated

[2] In what follows the value 'both' is atomic. Even if it encodes that a fact is both true and false, one cannot consider those cases separately without running into semantical problems.

[3] Recall that designated values are values that act as truth [11, 36].

values and further adjustments, does not affect the semantics of rule languages considered in [27, 28, 29]. It is introduced for conceptual simplification only.

Note that T_4 linearizes L_4 while preserving the order $\mathfrak{n} < \mathfrak{b}$ of A_4. This ordering reflects the intuition that, at a given stage of information acquisition,

$t < t'$ indicates that t' "contains more truth" than t does.

Truth tables for connectives are provided in Table 1. Observe that they reflect the same principles for $\neg, \wedge, \vee$ as in (2.1) except that we use order T_4 rather than L_4. However, under T_4 disjunction and conjunction conform to principles (3.1) and (3.2). The same truth tables have been considered in [27, 28, 29].

TABLE 1. Truth tables for $\wedge$, $\vee$, $\rightarrow$ and $\neg$.

$\wedge$	$\mathfrak{f}$	$\mathfrak{n}$	$\mathfrak{b}$	$\mathfrak{t}$
$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{f}$
$\mathfrak{n}$	$\mathfrak{f}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$
$\mathfrak{b}$	$\mathfrak{f}$	$\mathfrak{n}$	$\mathfrak{b}$	$\mathfrak{b}$
$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{n}$	$\mathfrak{b}$	$\mathfrak{t}$

$\vee$	$\mathfrak{f}$	$\mathfrak{n}$	$\mathfrak{b}$	$\mathfrak{t}$
$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{n}$	$\mathfrak{b}$	$\mathfrak{t}$
$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{n}$	$\mathfrak{b}$	$\mathfrak{t}$
$\mathfrak{b}$	$\mathfrak{b}$	$\mathfrak{b}$	$\mathfrak{b}$	$\mathfrak{t}$
$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$

$\rightarrow$	$\mathfrak{f}$	$\mathfrak{n}$	$\mathfrak{b}$	$\mathfrak{t}$
$\mathfrak{f}$	$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$
$\mathfrak{n}$	$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$	$\mathfrak{t}$
$\mathfrak{b}$	$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{t}$	$\mathfrak{f}$
$\mathfrak{t}$	$\mathfrak{f}$	$\mathfrak{f}$	$\mathfrak{t}$	$\mathfrak{t}$

	$\neg$
$\mathfrak{f}$	$\mathfrak{t}$
$\mathfrak{n}$	$\mathfrak{n}$
$\mathfrak{b}$	$\mathfrak{b}$
$\mathfrak{t}$	$\mathfrak{f}$

EXAMPLE 3.1 (Example 2.2 continued)
Consider $lr(X) \vee bg(X)$ rather than $lr(X) \vee_{L_4} bg(X)$. Recall that the service has been told three facts: $lr(s_1), lr(s_2), \neg lr(s_2)$. The result of $lr(s_1) \vee bg(s_1)$ is now $\mathfrak{t}$ and the result for $lr(s_2) \vee bg(s_2)$ is $\mathfrak{b}$, making perfect intuitive sense in both cases. ◁

A discussion of $\rightarrow$ can still be helpful. Observe that implication can only be $\mathfrak{t}$ or $\mathfrak{f}$. It reflects the following principles:

- truth or falsity of a given proposition can only be deduced on the basis of true assumptions;
- on the basis of contradictory premises one should derive only contradictory conclusions;
- one should not derive conclusions on the basis of false or unknown premises.

(3.4)

The first and the third principle of (3.4) are rather obvious and accepted in monotonic reasoning. The following example justifies the second one.

EXAMPLE 3.2
Consider implication:

safe(location) $\rightarrow$ *stay_in(location)*. (3.5)

If *safe* is $\mathfrak{b}$ and *stay_in(location)* is $\mathfrak{t}$, we do not really want (3.5) to be $\mathfrak{t}$. We rather prefer that it is $\mathfrak{f}$, so that the rule engine will have to modify the conclusion *stay_in(location)* to $\mathfrak{b}$ not letting the agent believe that that it should stay in the *location*. ◁

Observe that the implication we use generalizes the modus ponens rule in the following sense:

$$\mathcal{I}^v(A) = \mathfrak{t},\ \mathcal{I}^v(A \rightarrow B) = \mathfrak{t} \ \vdash\ \mathcal{I}^v(B) \in \{\mathfrak{t}, \mathfrak{b}\}\,; \tag{3.6}$$

$$\mathcal{I}^v(A) = \mathfrak{b},\ \mathcal{I}^v(A \rightarrow B) = \mathfrak{t} \ \vdash\ \mathcal{I}^v(B) = \mathfrak{b}. \tag{3.7}$$

Summarizing (3.6) and (3.7), we conclude that modus ponens actually holds, i.e, the following rule is valid:

$$A,\ A \to B \ \vdash\ B. \tag{3.8}$$

Observe also that our implication provides semantics for rules. Therefore,

- it reflects fusing knowledge from various rules, i.e., acts according to K_4 rather than T_4;
- it reflects the intuition behind rules in logic, where one derives conclusions on the basis of designated values only.

To see that $\to$ indeed reflects the above properties consider K_4 with truth values $\mathbf{f}$ and $\mathbf{n}$ unified:

$$\mathbf{n} = \mathbf{f} \prec \mathbf{t} \prec \mathbf{b}. \tag{3.9}$$

Then $A \to B$ is $\mathbf{t}$ if and only if $A \preceq B$, where $\preceq$ is the reflexive closure of ordering (3.9).

Let us emphasize that:

- when one restricts truth values to $\{\mathbf{t}, \mathbf{f}\}$ then connectives defined in Table 1 become equivalent to their counterparts in classical propositional logic;
- when one restricts truth values to $\{\mathbf{t}, \mathbf{n}, \mathbf{f}\}$ or to $\{\mathbf{t}, \mathbf{b}, \mathbf{f}\}$ then conjunction, disjunction and negation become respectively their counterparts in Kleene three-valued logic K_3 with the third (non-classical) value meaning *undetermined* and in Priest logic P_3 [35], where the third value receives the meaning *paradoxical*.[4]

REMARK 3.3

1. Note that implication of P_3 is not quite satisfactory. As indicated in IV.4 of [35], implication of P_3 causes problems with modus ponens, modus tollens and reduction to absurdity. With the implication defined in Table 1 we have modus ponens. However, modus tollens and reduction to absurdity are not valid either. Namely:
   - a counterexample to modus tollens ($A \to B, \neg B \vdash \neg A$) can be $A = \mathbf{n}, B = \mathbf{f}$;
   - a counterexample to the reduction to absurdity ($(\neg A) \to \mathbf{f} \vdash A$) can be $A = \mathbf{n}$.

   On the other hand, when only truth values $\{\mathbf{t}, \mathbf{b}, \mathbf{f}\}$ are allowed, modus ponens and the reduction to absurdity are valid, so the implication we propose behaves better than the implication of P_3.
2. It is important to emphasize that our implication is tailored to deal with literals in conclusions, which precisely reflects syntax of rules. It is not meant as a general implication. For example, consider formula $A \to (A \vee B)$. When A takes the value $\mathbf{b}$ and B takes the value $\mathbf{t}$, this implication reduces to $\mathbf{b} \to (\mathbf{b} \vee \mathbf{t})$, i.e., to $\mathbf{f}$. On the other hand, it is reasonable to expect that $A \to (A \vee B)$ is $\mathbf{t}$. Of course, this phenomenon does not occur when conclusion is just a literal. ◁

DEFINITION 3.4

Let $\mathcal{C}$ be a set of constants and v be a *valuation* assigning constants to variables. By a *first-order interpretation over* $\mathcal{C}$ we mean any set $\mathcal{I}$ of ground literals such that for any constant

[4] The only difference between K_3 and P_3 is that only *true* is designated in K_3, while in P_3 both *true* and *paradoxical* are.

c occurring in literals in $\mathcal{I}$, $c \in \mathcal{C}$. The truth value of formula A in interpretation $\mathcal{I}$ under valuation v, denoted by $\mathcal{I}^v(A)$, is defined by:[5]

$$\begin{array}{lcl}
\mathcal{I}^v(\ell) & \stackrel{\text{def}}{=} & \mathcal{I}(v(\ell)), \text{ where } \ell \text{ is a literal;} \\
\mathcal{I}^v(\neg B) & \stackrel{\text{def}}{=} & \neg\mathcal{I}^v(B); \\
\mathcal{I}^v(B \circ C) & \stackrel{\text{def}}{=} & \mathcal{I}^v(B) \circ \mathcal{I}^v(C) \text{ for } \circ \in \{\wedge, \vee, \rightarrow\}; \\
\mathcal{I}^v(\forall X B(X)) & \stackrel{\text{def}}{=} & \min\{\mathcal{I}^v(B(c)) \mid c \in \mathcal{C}\}; \\
\mathcal{I}^v(\exists X B(X)) & \stackrel{\text{def}}{=} & \max\{\mathcal{I}^v(B(c)) \mid c \in \mathcal{C}\},
\end{array}$$

where the meaning of $\circ$ at righthand sides of equalities is provided in Table 1; and $\min$, $\max$ are operations of minimum and maximum w.r.t. ordering T_4 defined by (3.3). ◁

Since $\{\mathtt{t}, \mathtt{b}\}$ are designated values, we have the following definition of models.

DEFINITION 3.5
Let a set of constants, $\mathcal{C}$, be given. A set of ground literals $\mathcal{I}$ with constants in $\mathcal{C}$ is a *model of a set of formulas* G iff for each formula $A \in G$ and any valuation v mapping variables into $\mathcal{C}$, we have that $\mathcal{I}^v(A) \in \{\mathtt{t}, \mathtt{b}\}$. ◁

4 The 4QL Query Language

Let us recall some basic aspects of 4QL, a DATALOG$^{\neg\neg}$-like rule language introduced and investigated in [27, 28].[6] 4QL supports a modular and layered architecture, and provides a tractable framework for many forms of rule-based reasoning both monotonic and nonmonotonic. As the underpinning principle, openness of the world is assumed, which may lead to the lack of information. Negation in rule heads may lead to inconsistencies. To reduce the unknown/inconsistent zones, *modules* and *external literals* provide means for:

- the application-specific disambiguation of inconsistent information;
- the use of Local Closed World Assumption (thus also Closed World Assumption, whenever needed);
- the implementation of various forms of nonmonotonic and defeasible reasoning.

The semantics is based on the logic defined in Section 3.

To express nonmonotonic/defeasible rules we need the concept of modules and external literals. We skip this part here, as it will be explored and generalized in Section 5.

DEFINITION 4.1
By a *rule* we mean any expression of the form:

$$\ell \leftarrow b_{11}, \ldots, b_{1i_1} \mid \ldots \mid b_{m1}, \ldots, b_{mi_m}. \tag{4.1}$$

where ℓ is a literal, $\{b_{11}, \ldots, b_{1i_1}, \ldots, b_{m1}, \ldots, b_{mi_m}\}$ is a (possibly empty) set of literals; and ',' and '|' abbreviate conjunction and disjunction, respectively.

Literal ℓ is called the *head* of the rule and the expression at the righthand side of '$\leftarrow$' in (4.1) is called the *body* of the rule. If ϱ is a rule of the form (4.1) then $head(\varrho) \stackrel{\text{def}}{=} \ell$.

[5] Note that $\mathcal{I}(v(A))$ has been introduced in Definition 2.1. Additionally, we extend v to cover all formulas in the standard way.

[6] For related material and an experimental interpreter designed by P. Spanily see `4ql.org`.

A rule without variables is called a *ground rule*. A ground rule with the empty body is called a *fact*.

A 4QL program is a finite set of rules, where it is assumed that for any ground literal ℓ, the set of ground instances of rules of the program can contain at most one rule with ℓ as its head. ◁

For facts we use notation 'ℓ.' rather than '$\ell \leftarrow$.'

REMARK 4.2
In what follows we often assume that the considered rules are ground. It will always be done without loss of generality: whenever we shall restrict considerations to ground rules, the context will always allow to replace non-ground rules with their ground instances. Note also that this replacement does not affect data complexity of considered problems as in this case the size of considered expressions is assumed to be constant. ◁

Since we consider databases, from now on we assume that all database domains are finite. Also, the considered sets of constants are always assumed to be finite.

DEFINITION 4.3
Let a set of constants $\mathcal{C}$ be given. A set of ground literals $\mathcal{I}$ with constants in $\mathcal{C}$ is a *model of a set of rules* S iff for any valuation v mapping variables into constants in $\mathcal{C}$:

1. for each fact ℓ. we have that $\mathcal{I}(v(\ell)) \in \{\mathbf{t}, \mathbf{b}\}$;
2. for each rule $\big(\ell \leftarrow b_{11}, \ldots, b_{1i_1} \mid \ldots \mid b_{m1}, \ldots, b_{mi_m}.\big) \in S$ we have that
$\mathcal{I}\big(v\big((b_{11} \wedge \ldots \wedge b_{1i_1}) \vee \ldots \vee (b_{m1} \wedge \ldots \wedge b_{mi_m}) \rightarrow \ell\big)\big) = \mathbf{t}$. ◁

The semantics of 4QL is defined by well-supported models generalizing the idea of [21]. Namely, querying a 4QL program depends on querying the well-supported model of this program. Intuitively, a model is *well-supported* if all derived literals are supported by a reasoning grounded in facts. We again skip a formal definition of well-supportedness of [27, 28], as its alternative characterization will be provided in Definition 4.8.

It appears that for any set of rules there is a unique well-supported model and it can be computed in deterministic polynomial time. For details see [27, 28]. Let us only illustrate the use of 4QL rules on an example based on a one from [27].

EXAMPLE 4.4
Let R be the following set of rules:

$$
\begin{array}{lcl}
r & \leftarrow & s \mid t\,. \\
t & \leftarrow & r\,. \\
\neg s & \leftarrow & t\,. \\
s\,. &&
\end{array}
$$

A minimal model of R is $\{s, \neg s, r, t\}$. However, the only fact, s, has in this model the value $\mathbf{b}$ so there are no facts supporting the truth of r and t in this model. The intuitively correct model for R is the well-supported model $\{s, \neg s, r, \neg r, t, \neg t\}$. ◁

To make our presentation as self-contained as possible, let us now recall an algorithm of [27] for computing the unique well-supported model for a given set of rules. Its correctness is proved in [28].

We will need the following definition.

DEFINITION 4.5
Let S be a set of rules.

1. If $\neg\ell$ is a negative literal then by its *duplicate* we understand a fresh literal representing its positive part, for simplicity denoted by ℓ'.
2. By $Pos(S)$ we understand the DATALOG program obtained from S by replacing each negative literal $\neg\ell$ of S by its duplicate ℓ'. ◁

Input: a set of ground rules S.
Output: the unique well-supported model $\mathcal{I}^S$ for S.

1. (*finding basic inconsistencies*):
   (a) compute the least Herbrand model $\mathcal{I}_0^S$ of $Pos(S)$
   (b) let $\mathcal{I}_1^S \stackrel{\text{def}}{=} \{\ell, \neg\ell \mid \ell, \ell' \in \mathcal{I}_0^S\}$
2. (*finding potentially true literals*):
   (a) let $S' = \{\varrho \mid \varrho \in S \text{ and } \mathcal{I}_1^S\big(head(\varrho)\big) \neq \mathfrak{b}\}$
   (b) set $\mathcal{I}_2^S$ to be the least Herbrand model for $Pos(S')$
   with literals ℓ' substituted by $\neg\ell$
3. (*reasoning with inconsistency*):
   (a) define the following transformation Φ^S on interpretations:

$$\Phi^S(\mathcal{I}) \stackrel{\text{def}}{=} \mathcal{I} \cup \big\{\ell, \neg\ell \mid \text{there is a rule } [\ell \leftarrow \beta_1 \vee \ldots \vee \beta_m] \in S \quad (4.2)$$
$$\text{such that } \exists k \in \{1, \ldots, m\}[\mathcal{I}(\beta_k) = \mathfrak{b}] \quad (4.3)$$
$$\text{and } \neg\exists n \in \{1, \ldots, m\}[\,(\mathcal{I}_2^S - \mathcal{I})(\beta_n) = \mathfrak{t}\,]\big\}. \quad (4.4)$$

   The transformation Φ^S is monotonic (see [28]).
   Denote by $\mathcal{I}_3^S$ the fixpoint of Φ^S obtained by iterating Φ^S on $\mathcal{I}_1^S$, i.e.,
   $\mathcal{I}_3^S = \bigcup_{i \in \omega} (\Phi^S)^i(\mathcal{I}_1^S)$
   (b) set $\mathcal{I}^S = \mathcal{I}_2^S \cup \mathcal{I}_3^S$.

FIG. 2. Algorithm computing the well-supported model for a given set of ground rules [27].

The algorithm of [27] is presented in Figure 2. Let us now observe some of its properties leading to an alternative definition of well-supportedness.

The following lemmas are proved in [28].

LEMMA 4.6
If $\ell \in \mathcal{I}_1^S$ then the value of ℓ is $\mathfrak{b}$ in every model of S. ◁

LEMMA 4.7
If for a well-supported model $\mathcal{M}$ of S we have $\mathcal{M}(\ell) = \mathfrak{t}$ then $\ell \in \mathcal{I}_2^S$. ◁

Let us now provide a definition of well-supported models alternative to the one given in [27, 28].

DEFINITION 4.8
Let S be a set of rules. By the well-supported model of S we understand the least model of S containing $\mathcal{I}_1^S$ and such that all literals with the value $\mathfrak{t}$ in this model are contained in $\mathcal{I}_2^S$. ◁

The above definition is not quite "declarative" as referring to interpretations computed by the algorithm shown in Figure 2. On the other hand, using Lemmas 4.6 and 4.7 one can reformulate them in terms of least Herbrand models for DATALOG programs. Such a reformulation is, however, much less readable than Definition 4.8.

The following theorem justifies Definition 4.8.

THEOREM 4.9
Algorithm given in Figure 2 computes the well-supported model defined in Definition 4.8.

PROOF (Sketch)
Observe that $\mathcal{I}^S$ computed by the algorithm is the union of $\mathcal{I}_2^S \cup \mathcal{I}_3^S$. By definition of $\mathcal{I}_3^S$ we have that $\mathcal{I}_1^S \subseteq \mathcal{I}_3^S$, so also $\mathcal{I}_1^S \subseteq \mathcal{I}^S$.

Note that $\Phi(\mathcal{I})$ "corrects" the interpretation $\mathcal{I}$ by adding literals to assign $\mathfrak{b}$ to certain heads of rules when satisfiability of a rule is violated. Such a violation happens when the value of the body is $\mathfrak{b}$ and the value of its head is not $\mathfrak{b}$. When bodies of rules are evaluated, true literals are taken from $\mathcal{I}_2^S - \mathcal{I}$. Note that $\mathcal{I}_3^S$ is the least fixpoint of Φ, so the constructed model is the minimal one satisfying Definition 4.8. ◁

EXAMPLE 4.10
Let R be the set of rules defined in Example 4.4. Then $\mathcal{I}_1^R = \{s(a), \neg s(a)\}$. Therefore, $\mathcal{I}_2^R = \emptyset$, so there are no literals obtaining the value $\mathfrak{t}$ in the well-supported model of R. ◁

A 4QL query is any first-order formula. Given a well-supported model $\mathcal{M}$ for a 4QL program, the meaning of a query, say $A(\bar{x})$, is the set of tuples consisting of constants which, assigned to free variables $\bar{x}$ make the formula $\mathfrak{t}$ or $\mathfrak{b}$ in $\mathcal{M}$.

5 From 4QL to 4QL+

In this section we discuss extensions to 4QL which will lead to 4QL$^+$. In the rest of the paper we will not fix a particular syntax for rules and programs, using directly the corresponding propositional connectives and quantifiers with the semantics defined by Definition 3.4. Additionally, for notational convenience, we will use connective $\leftarrow$ defined by

$$A \leftarrow B \stackrel{\text{def}}{\equiv} B \rightarrow A.$$

5.1 Modules

Two substantial features of 4QL are modules and external literals. Modules, known from programming languages, are meant to be a representational tool to model different information sources. Also in 4QL rules can be grouped in modules. Modules can ask other modules about the values of literals. This is achieved by using external literals. However, a layered architecture is required [27, 28]:

- modules are organized in layers;
- a module m can query another module n when n is in a strictly lower layer than m.

The semantics of modules is given by well-supported models constructed for such modules starting from the lowest layer until the well-supported model of a given module is computed.

Modules and external literals are substantial for expressing nonmonotonic rules, e.g., closing the world, allowing lightweight default or defeasible reasoning, etc. (see examples in Section 5.3).

Observe that 4QL has some limitations:

- a literal can be a head of at most one rule;
- a layered architecture for modules is required;
- queries to modules are restricted to literals.

Our goal is to define query language 4QL$^+$, substantially relaxing these restrictions. Both 4QL and 4QL$^+$ will have the same expressive power with queries in PTIME and capturing PTIME. However, 4QL$^+$ will allow for more compact queries, easier to read and understand, e.g., allowing arbitrary sequences of quantifiers in rule bodies and first-order queries to external modules.

In the rest of the paper we do not provide any particular syntax to define modules. We will simply assume that heads of rules indicate modules in which such rules are embedded. We shall define this concept formally in subsequent sections. The following example provides some initial intuitions.

EXAMPLE 5.1
Consider the following rules:

$$\neg m.r(X) \leftarrow m.s(X,Y) \wedge n.r(X) \wedge m.r(Y). \tag{5.1}$$
$$n.r(X) \quad \leftarrow n.t(X). \tag{5.2}$$

Rule (5.1) belongs to module m, which is indicated by module name m in the head of the rule. Similarly, rule (5.2) belongs to module n. External literals $n.r(X), m.r(Y)$ appearing in the body of rule (5.1) indicate that the value of $r(X)$ should be told to m by module n and the values of $r(Y)$ and $s(X,Y)$ should be taken from module m. Rule (5.2) refers only to the relation $n.t$ which is internal to n. ◁

5.2 The Same Head in More than One Rule

The first extension to 4QL we propose is to allow literals to be heads of more than one rule. To illustrate the intended semantics, consider the following two rules:

$$\ell \leftarrow \beta_1. \tag{5.3}$$
$$\ell \leftarrow \beta_2. \tag{5.4}$$

Let us emphasize that in our four-valued logic (with the semantics of connectives defined in Table 1), the conjunction of rules (5.3) and (5.4) is not equivalent to the rule:

$$\ell \leftarrow \beta_1 \vee \beta_2. \tag{5.5}$$

Namely, in the case when one of β_1, β_2 is $\mathfrak{t}$ and the other is $\mathfrak{b}$ the conjunction of rules (5.3) and (5.4) results in $\ell = \mathfrak{b}$ while rule (5.5) results in $\ell = \mathfrak{t}$. Disjunction within the body of rules is the disjunction w.r.t. truth ordering T_4 defined by (3.3). When we have two (or more) rules with the same head then we treat them as different sources providing information about the head, so ordering in which $\mathfrak{t} < \mathfrak{b}$, as in lattice A_4, is adequate to fuse the results. More precisely, consider the following order, which we denote by K_4 (see also Figure 1):

$$\mathfrak{n} < \mathfrak{f} < \mathfrak{t} < \mathfrak{b}. \tag{5.6}$$

Observe that K_4 is a linearization of A_4, where ordering $\mathfrak{f} < \mathfrak{t}$ reflects the classical ordering (as well as T_4).

A formal meaning of rules with the same head is now given by:

$$\big((p \to r) \wedge (q \to r)\big) = \big((p \vee_{K_4} q) \to r\big), \tag{5.7}$$

where $\vee_{k_4}$ stands for the disjunction w.r.t. ordering K_4 and $=$ denotes equality on truth values of formulas in the sense that $A = B$ if for any valuation assigning truth values to propositions, the truth value of A is the same as the truth value of B.

Note that, in some cases, algorithms shown in this paper do not compute well-supported models when the same head may appear in more than one rule. An algorithm dealing with such cases is being investigated.

5.3 Multisource Formulas

In [27, 28] external literals have been defined as follows.

Let Mod be a finite set of *module names* with a strict partial ordering $\prec$ on Mod. The ordering reflects a layered architecture in the sense that when $m_1 \prec m_2$ then we say that m_1 is in a strictly lower layer than m_2.

DEFINITION 5.2

1. An *external literal* is an expression of one of the forms: $m.r, \neg m.r, m.r \in T, \neg m.r \in T$, where:
   - $m \in Mod$ is a module name and r is a positive literal; m is called the *reference module* of the external literal. If r does not appear in the module m then the value of $m.r$ is $\mathfrak{n}$;
   - $T \subseteq \{\mathfrak{t}, \mathfrak{b}, \mathfrak{n}, \mathfrak{f}\}$.[7]
2. An external literal may only appear in rule bodies of a module m, provided that its reference module is in a strictly lower layer than m. ◁

The semantics of modules and external literals can easily be defined by assuming that:

- each module operates on its "local" literals, accessing "foreign" literals only via external literals;
- external literals, when used in a given module, are fully defined in modules in lower layers;
- truth values assigned to external literals, once computed, do not change.

Under these assumptions, each external literal occurring in a module m has a fixed truth value determined at lower levels. The semantics of m is now defined as the unique well-supported model of the program m' obtained from m by replacing each external literal of m by the respective constant $\mathfrak{f}$, $\mathfrak{n}$, $\mathfrak{b}$ or $\mathfrak{t}$ denoting the truth value of this literal. In consequence, we extend the definition of facts as follows.

DEFINITION 5.3

A *fact* is either a fact, as defined in Definition 4.1, or a ground external literal of the form $m.r(\bar{a}) \in T$ or $\neg m.r(\bar{a}) \in T$, assuming that the value of $r(\bar{a})$ (respectively $\neg r(\bar{a})$) in the well-supported model $\mathcal{I}^m$ of m is in the set of truth values T, i.e., $\mathcal{I}^m(r(\bar{a})) \in T$ (respectively, $\mathcal{I}^m(\neg r(\bar{a})) \in T$). ◁

Note that one can consider formulas rather than external literals. This makes queries to external modules as powerful as SQL queries. We will call such formulas *multisource formulas*

[7] The intended meaning of $A.R \in T$ is that the truth value of $a.r$ is in the set T.

TRUTHVALUE::=	$\mathbf{t} \mid \mathbf{b} \mid \mathbf{n} \mid \mathbf{f}$
LISTOFTRUTHVALUES::=	TRUTHVALUE | TRUTHVALUE, LISTOFTRUTHVALUES
MULTISOURCEFORMULA::=	MODULENAME.CLASSICALLITERAL |
	¬MULTISOURCEFORMULA |
	MULTISOURCEFORMULA ∨ MULTISOURCEFORMULA |
	MULTISOURCEFORMULA ∧ MULTISOURCEFORMULA |
	∃ VARIABLE (MULTISOURCEFORMULA) |
	∀ VARIABLE (MULTISOURCEFORMULA) |
	MODULENAME.[MULTISOURCEFORMULA] |
	MODULENAME.[MULTISOURCEFORMULA]∈{LISTOFTRUTHVALUES}

FIG. 3. Syntax of multisource formulas.

to emphasize their role as queries aggregating information from possibly multiple sources. Syntax of multisource formulas is given in Figure 3. It is, however, required that any literal appears within a scope of a module, where the scope is defined as follows.[8]

DEFINITION 5.4
The *scope* of m in formula $m.[A]$ is the formula A. By the *reference module of a literal* ℓ we understand the occurrence of m such that ℓ is in the scope of m and there is no n appearing in the scope of m with ℓ being in the scope of n. ◁

REMARK 5.5
To avoid unnecessary repetitions of module names we shall use some simplifications. First, when the context is clear, we omit brackets '[' and ']' in formulas of the form $m.[A]$. Next, when A is a negative literal then we write $\neg m.A$ rather than $m.[\neg A]$. Also, when the reference module of a literal is known from the context, we omit the module name within the literal. Finally, we often remove redundant occurrences of module names. ◁

EXAMPLE 5.6
Formula $m.\big[\exists X(r(X) \wedge s(X)) \wedge n.[s(a) \vee r(a)]\big]$ is to be understood as:
$\exists X\big(m.r(X) \wedge m.s(X)\big) \wedge \big(n.s(a) \vee n.s(a)\big)$. ◁

We are now in position to define the semantics of multisource formulas. Intuitively, the semantics is given by interpretations attached to module names.

DEFINITION 5.7
Let $\mathcal{C}$ be a set of constants, GL be the set of ground literals with constants in $\mathcal{C}$; and let $\Gamma : Mod \longrightarrow 2^{GL}$ be a mapping assigning sets of ground literals to module names. Let A be a formula with free variables $\bar{X}$. The semantics of multisource formulas is defined as in Definition 3.4,[9] together with the following clauses, where v is a valuation of variables:

- $\Gamma^v(\ell) \stackrel{\text{def}}{=} \mathcal{I}^v(\ell)$, where ℓ is a literal and $\mathcal{I} = \Gamma(m)$ with m being the reference module of ℓ;
- $\Gamma^v(A(X) \in T) \stackrel{\text{def}}{=} \begin{cases} \mathbf{t} & \text{when } \Gamma^v(A(X)) \in T; \\ \mathbf{f} & \text{otherwise.} \end{cases}$ ◁

The meaning of multisource formulas generalizes the meaning of external literals. However, we substantially relax the layered architecture, as explained in Section 5.5. Note that

[8] The idea behind the definition of multisource formulas is that there should be unambiguous references to modules indicating where subformulas should be evaluated.

[9] However, rather than a single interpretation, we consider interpretations provided by Γ.

multisource formulas provide a shift in perspective. Namely, when module m asks module n about values of literals and then computes the value of a formula A involving these values, then A is computed by m. When $n.A$ is the query, then n computes the answer and tells m the result. In many applications this is more natural. Of course, m may simulate the same computation asking n about values of all literals appearing in A and then determine the result itself. However, this is less readable. Moreover, assuming a natural evaluation strategy, where $n.A$ indicates that A is to be computed by n, this simulation may be less efficient, since many queries about literals have to be asked and answered comparing to just one query A.

EXAMPLE 5.8
Let *danger*(X) express that there is a potentially dangerous situation in region X, $sub(X, Y)$ express that region X is a subregion of region Y and *high*(X, Z) express that the value of attribute Z in region X is high. Consider the following formula:

$$m.\mathit{danger}(X) \leftarrow n.[\exists Y(sub(Y, X) \wedge \mathit{high}(Y, \mathit{pollution}) \wedge \mathit{high}(Y, \mathit{temperature}))] \in \{\mathbf{t}, \mathbf{b}, \mathbf{n}\}.$$

The formula states that m finds the situation in region X dangerous when the value of formula

$$\exists Y(sub(Y, X) \wedge \mathit{high}(Y, \mathit{pollution}) \wedge \mathit{high}(Y, \mathit{temperature})),$$

evaluated by module n, is in $\{\mathbf{t}, \mathbf{b}, \mathbf{n}\}$. That is, m concludes that there is a danger in region X if n tells m that there is a subregion of X, where values of pollution and temperature are high or n has inconsistent or no information whether these values are high.

Note that computing the same answer in 4QL may require asking queries about literals $sub(Y, X)$, $high(Y, \mathit{pollution})$, $high(Y, \mathit{temperature})$ for all Y being subregions of X. In distributed systems communication usually takes its time. In such circumstances there can be a considerable loss in performance of the system. The underlying evaluation strategy used by the database engine should address this point. ◁

5.4 Querying Information Sources

Multisource formulas are meant to be used to query databases. From the point of view of the current paper, databases are defined by means of interpretations, i.e., sets of literals. For 4QL and 4QL$^+$ such sets are constructed as well-supported models. However, they can also be provided by sources not necessarily using 4QL or 4QL$^+$. Note that Definition 5.7 is independent of well-supported models, as Γ assigns to module names arbitrary sets of ground literals.

In standard databases a query returns a set of tuples satisfying the query. In deductive databases queries are expressed by formulas [1, 37]. In our approach, queries are expressed by multisource formulas. If a query contains no free variables then it returns a truth value. This case is already covered by Definition 5.7. Consider then a query A with free variables $\bar{X}$. Note that the set of designated values is $\mathbf{t}, \mathbf{b}$. Therefore we have to return tuples $\bar{a}$ such that the value of $A(\bar{a})$ is in $\{\mathbf{t}, \mathbf{b}\}$. It is, however, necessary to distinguish tuples making $A(\bar{a}) = \mathbf{t}$ from those making $A(\bar{a}) = \mathbf{b}$. It is also useful to distinguish tuples making $A(\bar{a}) = \mathbf{f}$ or $A(\bar{a}) = \mathbf{n}$. We then have the following definition.

DEFINITION 5.9
Let $\mathcal{C}$ and Γ be as in Definition 5.7. By an *annotated tuple* we mean an expression of the form $+\bar{a}$ or $-\bar{a}$, where $\bar{a}$ is a tuple of constants from $\mathcal{C}$.

For a multisource formula $A(\bar{X})$ the *meaning of query* expressed by $A(\bar{X})$ is the set of annotated tuples:[10]

$$\{+\bar{a} \mid \Gamma^v(A(\bar{a})) = \mathfrak{t}\} \cup \{-\bar{a} \mid \Gamma^v(A(\bar{a})) = \mathfrak{f}\} \cup \{+\bar{a}, -\bar{a} \mid \Gamma^v(A(\bar{a})) = \mathfrak{b}\}. \quad \triangleleft$$

Note that, given a set U of annotated tuples as defined in Definition 5.9, the truth value t of a query for a given tuple $\bar{a}$ can be computed in the standard way (cf. Definition 2.1):

$$t = \begin{cases} \mathfrak{t} & \text{when } +\bar{a} \in U \text{ and } -\bar{a} \notin U \\ \mathfrak{b} & \text{when } +\bar{a} \in U \text{ and } -\bar{a} \in U \\ \mathfrak{n} & \text{when } +\bar{a} \notin U \text{ and } -\bar{a} \notin U \\ \mathfrak{f} & \text{when } +\bar{a} \notin U \text{ and } -\bar{a} \in U. \end{cases}$$

EXAMPLE 5.10
Consider two information sources, m and n. Let $\mathcal{C} = \{a, b\}$ and:

$$\Gamma(m) = \{r(a,b), s(a), \neg s(a)\},\ \Gamma(n) = \{\neg r(a,b), s(b)\}.$$

Then query $m.[s(X)] \vee \neg n.[r(X,Y)]$ returns $\{+\langle a,b\rangle, +\langle a,a\rangle, -\langle a,a\rangle\}$. $\triangleleft$

If one does not want to select all annotated tuples, but only those for which the query returns a given set of truth values, multisource formulas of the form $m.[A(\bar{X})] \in T$ can be used. For example, $m.[s(X,Y)] \in \{\mathfrak{b}, \mathfrak{n}\}$ returns tuples $\langle a, b\rangle$ such that, in interpretation $\Gamma(m)$, we have $s(a,b) \in \{\mathfrak{b}, \mathfrak{n}\}$.

5.5 Multisource Rules

Let us now define multisource rules.

DEFINITION 5.11
By a *multisource rule* we understand any expression of the form $\ell \leftarrow \beta$, where:

1. ℓ, called the *head* of the multisource rule, is an external literal of the form $m.\ell'$ or $\neg m.\ell'$;
2. β, called the *body* of the multisource rule, is a multisource formula.

The *scope* of m appearing in a rule head is the whole rule. The head of a rule ϱ is denoted by $head(\varrho)$.

Multisource rules with empty bodies are called *facts* and are denoted by 'ℓ.' rather than '$\ell \leftarrow$.' . $\triangleleft$

REMARK 5.12
Note that the scope of rule $\ell \leftarrow \beta$ allows us to understand the rule as the multisource formula $m.[\ell \leftarrow \beta]$, where m is the module name occurring in ℓ. This also allows us to avoid redundant repetitions of module names in bodies of rules (as outlined in Remark 5.5). For example, using these conventions, a rule

$$\neg m.r(X) \leftarrow \forall Y \exists Z \big(s(X,Y,Z) \vee t(X) \vee n.r(Z)\big).$$

is understood as the following multisource formula:

$$m.[\neg m.r(X) \leftarrow \forall Y \exists Z \big(m.s(X,Y,Z) \vee m.t(X) \vee n.r(Z)\big)]. \quad \triangleleft$$

[10] Note that $A(\bar{a})$ contains no free variables, so v does not matter.

It is important to note that point 2. of Definition 5.2 is no longer required. It will be substantially relaxed (see Definition 5.15).

REMARK 5.13
Multisource formulas express queries to modules. They have the power of standard SQL queries. On the other hand, there are tractable queries not expressible in SQL [1]. Therefore, rather than first-order formulas one can allow a set of multisource rules to query modules. In such a case rather than $m.[A]$, where A is a multisource formula, one can allow $m.[S;\ell]$, where S is a set of rules and ℓ is a literal whose value is the result of the query. The meaning of such a query would then be:

> compute the well-supported model of S using facts and rules of m augmented with rules included in S and return the value of ℓ in this model as the resulting value.

To make the presentation as clear as possible, we do not consider this option in the current paper. Implementations of the proposed approach can allow to express such queries without losing tractability. ◁

It is natural to assume that modules represent agents' beliefs [19]. In agent systems one frequently assumes dialogues among agents. That is, an agent a_1 may ask a query to an agent a_2 and also be asked queries by a_2. Such situations are not directly supported by 4QL. Namely, to allow such queries, one has to distribute the set of rules among additional, perhaps artificial well-layered modules. In extreme cases, each rule would have to be placed in a separate module, making the whole set of rules less natural and less readable than necessary. Therefore we relax the definition of layering. In fact our new definition is close to that of stratification in DATALOG$^\neg$ (see, e.g., [1]).

To formally define the layered architecture we further assume that $\mathcal{C}$ is the set of all constants occurring in rules. To illustrate the main problem with multisource formulas as well as external literals, consider the following example.

EXAMPLE 5.14
Consider the following two rules:

$$m.r \leftarrow n.r \in \{\mathfrak{n}\}. \tag{5.8}$$

$$n.r \ \leftarrow m.r \in \{\mathfrak{n}\}. \tag{5.9}$$

There are many models of (5.8) and (5.9), e.g., $\{m.r\}$, $\{n.r\}$. Note that $\emptyset$ is not a model for these formulas. None of these models is intuitively better than the other. The situation is symmetric and no module is to be preferred. One could consider models $\{m.r, n.r\}$ or $\{m.r, \neg m.r, n.r, \neg n.r\}$ but these models are not grounded in any facts (in the sense of Definition 5.3), since $n.r \in \{\mathfrak{n}\}$ and $m.r \in \{\mathfrak{n}\}$ are both $\mathfrak{f}$ these models. ◁

In the following definition rather than requiring that modules are layered, we require that literals should allow a layered structure.

DEFINITION 5.15
Let S be a set of multisource rules and let L be the set of external literals appearing in S. We say that S is *admissible* iff there is a mapping $\kappa_S : L \longrightarrow \omega$ such that for every rule $\varrho \in S$,[11]

1. if ℓ is a literal appearing in the body of ϱ in the scope of a multisource formula of the form $n.[A]$ then $\kappa_S(head(\varrho)) \geq \kappa_S(\ell)$;

[11] Let us emphasize that the convention of Remark 5.12 applies.

2. if ℓ is a literal appearing in the body of ϱ in the scope of a multisource formula of the form $n.[A] \in T$, then $\kappa_S(head(\varrho)) > \kappa_S(\ell)$. ◁

The following example illustrates a substantial difference between the above definition and layering assumed in [27, 28].

EXAMPLE 5.16
Consider the following rules:

$$m.r \leftarrow n.r \in \{\mathfrak{n}\}.$$
$$n.s \leftarrow m.s \in \{\mathfrak{n}\}.$$

According to the definition of [27, 28], modules m and n cannot be layered. On the other hand, according to Definition 5.15, these rules are admissible. For example, one can define $\kappa(m.s) = \kappa(n.r) = 0$ and $\kappa(m.r) = \kappa(n.s) = 1$. This indicates that one can first compute values of $m.s$ and $n.r$ (both equal to $\mathfrak{n}$) and then values of $m.r$ and $n.s$ (both equal to $\mathfrak{t}$). ◁

Note that admissibility is a generalization of stratifiability of DATALOG$^\neg$ programs. The PTIME algorithm checking whether a set of rules of DATALOG$^\neg$ is stratifiable (see, e.g., [1]) can be applied to check admissibility. Therefore we have the following proposition.

PROPOSITION 5.17
For any set S of multisource rules, admissibility of S can be checked in deterministic polynomial time. ◁

6 The 4QL+ Query Language

Let us now define the 4QL$^+$ language.

DEFINITION 6.1
A *4QL$^+$ rule* is any multisource rule, as defined in Definition 5.11. A *4QL$^+$ program* is any admissible set of 4QL$^+$ rules. A *query to a 4QL$^+$ program* is any multisource formula. ◁

EXAMPLE 6.2
The following implication is a well-formed 4QL$^+$ query, where $m, n \in Mod$ are module names):

$$\forall X(m.[r(X) \vee \neg s(X)] \rightarrow \exists Y n.[n.r(Y) \wedge t(X, Y)]).$$

On the other hand, $r(a)$ is not a well-formed query, since $r(a)$ has no reference to a module, so is not a well-formed multisource formula. It becomes well-formed when a reference to a module is added like, e.g., $m.[r(a)]$. ◁

To construct an algorithm for computing well-supported models, let us first focus on the case when there is only one layer, i.e., there is $i \in \omega$ such that for any external literal ℓ labeling the head of a rule, $\kappa(\ell) = i$. Consider now a modification of algorithm provided in Figure 2, where the third phase (reasoning with inconsistency) is replaced by the algorithm given in Figure 4. Note also that least Herbrand models of phases 1 and 2 can be computed by evaluating suitable least fixpoints, so remain PTIME computable.

Note that definitions of $\mathcal{I}_1^S$ and $\mathcal{I}_2^S$ have the same meaning as before. Therefore, Definition 4.8 of well-supported models appears general enough to capture 4QL$^+$.

3'. (*reasoning with inconsistency*):
(a) **set** $\mathcal{I}^S = \mathcal{I}^S_1 \cup \mathcal{I}^S_2$;
(b) **while** there is a rule $[\ell \leftarrow \beta] \in S$ such that $\mathcal{I}^S(\beta) = \mathfrak{b}$ and $\mathcal{I}^S(\ell) \neq \mathfrak{b}$
do: add $\{\ell, \neg\ell\}$ to $\mathcal{I}^S$.

FIG. 4: Phase 3 of the algorithm computing the well-supported model for single-layered set of multisource rules S.

The new algorithm can easily be extended to cover any finite number of layers. Assume that there are layers $0, \ldots, k$. Then we can define the following algorithm for computing well-supported models for all modules:

for $i = 0, \ldots, k$ **do:**
compute the well-supported model for rules with heads ℓ such that $\kappa(\ell) = i$
substituting calls to external modules by respective truth values computed
in all layers j with $j < i$.

DEFINITION 6.3
The semantics of 4QL$^+$ rule $\ell \leftarrow \beta$ is defined by requiring that for all v assigning constants from $\mathcal{C}$ to variables, $\Gamma^v\big(\ell \leftarrow \beta\big) \in \{\mathfrak{t}, \mathfrak{b}\}$, where:

- $\mathcal{C}$ and Γ are as in Definition 5.7;
- for any m, $\Gamma(m)$ is the well-supported model for facts and relations with heads labeled by m. ◁

Note that each 4QL$^+$ program consists of a fixed number of layers, not changing during evaluation of queries. Well-supported models for modules of 4QL$^+$ programs are constructed layer by layer, where each layer is computed in time polynomial w.r.t. number of constants. Therefore we have the following theorem.

THEOREM 6.4
4QL$^+$ has PTIME data complexity. ◁

Since 4QL captures PTIME on ordered structures [28] and 4QL is a sublanguage of 4QL$^+$, we also have the following theorem.

THEOREM 6.5
4QL$^+$ captures PTIME queries on ordered structures. ◁

Let us finish this section by showing an example of the use of 4QL$^+$.

EXAMPLE 6.6
Consider a case study of [20] (Chapter 7.2) dealing with ecological diasters caused by specific poisons. The scenario, with some simplifications and adjustments, is the following: possible hazards are two kinds of poison, p_1 and p_2. They may also be explosive if they react with one another, which happens at high concentrations. Three attributes influence danger level: temperature T, pressure P and concentrations C_1, C_2 of poisons p_1, p_2 at a given location L. In [20] the main emphasis is put on cooperation between heterogeneous agents, and (sub)teams of agents. Here we will discuss fusing information from two sources: the first, a_1,

providing temperature and pressure and the second, a_2, providing concentration of poisons p_1, p_2, respectively. One could have the following simple rules:

$$\begin{array}{lcl}
a.\mathit{safe}(L, p_1) & \leftarrow & a_1.[\neg \mathit{high}(L, \mathit{temperature}) \wedge \neg \mathit{high}(L, \mathit{pressure})] \wedge \\
 & & a_2.[\neg \mathit{high}(L, \mathit{concentration}_1) \wedge \neg \mathit{high}(L, \mathit{concentration}_2)]. \\
a.\mathit{risky}(L, p_1) & \leftarrow & a_1.[\mathit{high}(L, \mathit{temperature}) \wedge \neg \mathit{high}(L, \mathit{pressure})] \vee \\
 & & a_2.[\mathit{high}(L, \mathit{concentration}_1) \wedge \mathit{high}(L, \mathit{concentration}_2)]. \\
a.\mathit{explosive}(L) & \leftarrow & a_2.[\mathit{high}(L, \mathit{concentration}_1) \wedge \mathit{high}(L, \mathit{concentration}_2)]. \\
a.\mathit{dangerous}(L) & \leftarrow & a_1.[\mathit{high}(L, \mathit{concentration}_1) \vee \mathit{high}(L, \mathit{concentration}_2)]. \\
\neg a.\mathit{dangerous}(L) & \leftarrow & a.[\mathit{safe}(L)].
\end{array}$$

It may happen that an information source fails to measure respective attributes, so that, e.g., $a.\mathit{explosive}(L)$ is $\mathfrak{n}$. Also, if information as to location L being in explosive state, is inconsistent, it is safe to assume that it actually is in such a state. One can express a rule addressing such situations in another module, say b, as:

$$b.\mathit{explosive}(L) \leftarrow a.\mathit{explosive}(L) \in \{\mathfrak{t}, \mathfrak{b}, \mathfrak{n}\}.$$

Of course, the above rule leads to nonmonotonicity of reasoning, but is very intuitive and natural. ◁

7 Conclusions

In the current paper we have presented a comprehensive technology for the construction of tractable advanced knowledge bases for agent systems. The technology is defined via the 4QL$^+$ query language extending 4QL [27, 28] and allowing one to express complex rules and queries. Rules allowed in 4QL$^+$ are far beyond those of DATALOG, DATALOG$^\neg$ and DATALOG$^{\neg\neg}$. The underlying semantics remains simple and guarantees tractability of computing queries. Moreover, all tractable queries can be expressed in 4QL$^+$.

Logical techniques used in the paper allow one to formulate agents' rules allowing to gather information from heterogeneous sources as well as for lightweight forms of nonmonotonic reasoning, including defeasible reasoning.

We intentionally avoided defining implementation-oriented syntax sticking to traditional logical notation. A programming language-like syntax for the use in applications is left for implementations of 4QL$^+$. In fact such implementations may extend the language in various directions. First, one can still strengthen queries as indicated in Remark 5.13. Also, some features important in practice of developing agent systems, like inheritance and more advanced encapsulation techniques from object-oriented languages as well as related agent techniques and constructs can then be incorporated.

Acknowledgments

I would like to thank editors of this special issue and anonymous reviewers for helpful comments and suggestions. This work has been supported by the Polish National Science Centre grant 2011/01/B/ST6/02769.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley Pub. Co., 1996.

[2] J.J. Alferes and L.M. Pereira. *Reasoning with Logic Programming*, volume 1111 of *LNCS*. Springer-Verlag, 1996.

[3] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.

[4] A.R. Anderson and N.D. Belnap. *Entailment: The Logic of Relevance and Necessity, Vol. I.* Princeton University Press, 1975.

[5] O. Arieli and A. Avron. The value of the four values. *Artificial Intelligence*, 102(1):97–141, 1998.

[6] A. Avron, J. Ben-Naim, and B. Konikowska. Logics of reasonable information sources. In F. Esteva, J. Gispert, and F. Manya, editors, *ISMVL*, pages 61–66. IEEE, 2010.

[7] N.D. Belnap. How a computer should think. In G. Ryle, editor, *Contemporary Aspects of Philosophy*, pages 30–55. Oriel Press, 1977.

[8] N.D. Belnap. A useful four-valued logic. In G. Epstein and J.M. Dunn, editors, *Modern Uses of Many Valued Logic*, pages 8–37. Reidel, 1977.

[9] J.-Y. Béziau, W. Carnielli, and D.M. Gabbay, editors. *Handbook of Paraconsistency*. College Publications, 2007.

[10] H.A. Blair and V.S. Subrahmanian. Paraconsistent logic programming. *Theor. Comput. Sci.*, 68(2):135–154, 1989.

[11] L. Bolc and P. Borowik. *Many-Valued Logics, 1. Theoretical Foundations*. Springer-Verlag, 1992.

[12] R.H. Bordini, L. Braubach, M. Dastani, A. El Fallah-Seghrouchni, J.J. Gmez-Sanz, J. Leite, G.M.P. O'Hare, A. Pokahr, and A. Ricci. A survey of programming languages and platforms for multi-agent systems. *Informatica*, 30:33–44, 2006.

[13] G. Brewka. *Non-Monotonic Reasoning: Logical Foundations of Commonsense*. Cambridge University Press, 1991.

[14] M. Cadoli and M. Schaerf. A survey on complexity results for non-monotonic logics. *Journal Logic Programming*, 17:127–160, 1993.

[15] C.V. Damásio and L.M. Pereira. A survey of paraconsistent semantics for logic programs. In D. M. Gabbay and P. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 2, pages 241–320. Kluwer, 1998.

[16] S. de Amo and M.S. Pais. A paraconsistent logic approach for querying inconsistent databases. *International Journal of Approximate Reasoning*, 46:366–386, 2007.

[17] D. Dubois. On ignorance and contradiction considered as truth-values. *Logic Journal of the IGPL*, 16(2):195–216, 2008.

[18] D. Dubois. Author's response to Wansing and Belnap's *generalized truth-values*. *Logic Journal of the IGPL*, 18(6):936–940, 2010.

[19] B. Dunin-Kęplicz and A. Szałas. Epistemic profiles and belief structures. In *Proc. AMSTA 2012: Agents and Multi-agent Systems Technologies and Applications*, number 7327 in LNCS, pages 360–369. Springer-Verlag, 2012.

[20] B. Dunin-Kęplicz and R. Verbrugge. *Teamwork in Multi-Agent Systems. A Formal Approach*. John Wiley & Sons, Ltd., 2010.

[21] F. Fages. Consistency of Clark's completion and existence of stable models. *Methods of Logic in Computer Science*, 1:51–60, 1994.

[22] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning About Knowledge*. The MIT Press, 2003.

[23] J. Fox. Motivation and demotivation of a four-valued logic. *Notre Dame Journal of Formal Logic*, 31(1):76–80, 1990.

[24] G. Gottlob. Complexity results for nonmonotonic logics. *J. Logic Computat.*, 2:397–425, 1992.

[25] J. Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.

[26] W. Łukaszewicz. *Non-Monotonic Reasoning - Formalization of Commonsense Reasoning*. Ellis Horwood, 1990.

[27] J. Małuszyński and A. Szałas. Living with inconsistency and taming nonmonotonicity. In O. de Moor, G. Gottlob, T. Furche, and A. Sellers, editors, *Datalog 2.0*, volume 6702 of *LNCS*, pages 334–398. Springer-Verlag, 2011.

[28] J. Małuszyński and A. Szałas. Logical foundations and complexity of 4QL, a query language with unrestricted negation. *Journal of Applied Non-Classical Logics*, 21(2):211–232, 2011.

[29] J. Małuszyński, A. Szałas, and A. Vitória. Paraconsistent logic programs with four-valued rough sets. In C.-C. Chan, J. Grzymala-Busse, and W. Ziarko, editors, *Proc. of 6th International Conference on Rough Sets and Current Trends in Computing (RSCTC 2008)*, volume 5306 of *LNAI*, pages 41–51, 2008.

[30] V.W. Marek and M. Truszczyński. *Nonmonotonic Logic*. Springer-Verlag, 1993.

[31] J.-J. Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Theoretical Computer Science*. Cambridge University Press, 1995.

[32] A.L. Nguyen and A. Szałas. Three-valued paraconsistent reasoning for Semantic Web agents. In P. Jedrzejowicz, N.T. Nguyen, R.J. Howlet, and L.C. Jain, editors, *4th International KES Symposium on Agents and Multi-Agent Systems*, volume 6070 of *LNAI*, pages 152–162, 2010.

[33] D. Nute. Defeasible logic. In D.M. Gabbay, C.J Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 353–395. Oxford University Press, 1994.

[34] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[35] G. Priest. The logic of paradox. *Journal of Philosophical Logic*, 8:219–241, 1979.

[36] N. Rescher. *Many-Valued Logic*. McGraw Hill, 1969.

[37] F. Sáenz-Pérez. DES: A deductive database system. *Electron. Notes Theor. Comput. Sci.*, 271:63–78, 2011.

[38] Y. Shramko and H. Wansing. *Truth and Falsehood: An Inquiry Into Generalized Logical Values*. Springer-Verlag, 2011.

[39] H. Wansing and N. Belnap. Generalized truth values.: A reply to Dubois. *Logic Journal of the IGPL*, 18(6):921–935, 2010.