Mathware & Soft Computing 1 (1994) 5-29

A Łukasiewicz Logic Based Prolog

Frank Klawonn & Rudolf Kruse
Department of Computer Science
University of Braunschweig
Braunschweig, Germany

Abstract

Prolog is a programming language based on a restricted subset of classical first order predicate logic. In order to overcome some problems of classical logic to handle imperfect human knowledge, we provide a formal framework for a Łukasiewicz logic based Prolog system. The use of Łukasiewicz logic with its connection to Ulam games enables us to deal with partial inconsistencies by interpreting the truth values as relative distance to contradiction.

We also present the software tool LULOG which is based on the theoretical results of this paper and can be seen as a Prolog system for many–valued logic. Applications of LULOG to an Ulam game and an example of reasoning with imperfect knowledge are also discussed.

1 Introduction

Classical logic provides a framework for the formulation and implementation of knowledge based systems. The programming language Prolog [6, 7] is based on a subset of first order predicate logic and thus a considerable number of artificial intelligence applications is implemented in Prolog. Neither Prolog nor classical logic are intended to cope with incomplete, contradictory, uncertain, or vague knowledge. But these phenomena are often encountered when knowledge based systems have to be designed.

Various theories for the treatment of one of these phenomena of imperfect knowledge have already been developed. Default logic [4] deals with incomplete information. Uncertainty can be handled by probabilistic [2] or possibilistic logic [10], whereas fuzzy logic [12, 17] is connected to vagueness. All these approaches to approximate reasoning are based on logical calculi, that will also provide the basis for the treatment of partially inconsistent knowledge for which the approach presented in this paper is suitable. Numerical approaches to approximate reasoning [11, 18] are not considered here. An overview on logical approaches to approximate reasoning can be found in [8, 29].

The idea to extend Prolog to be able to carry out approximate reasoning is proposed by many authors [3, 13, 19, 21, 22, 31]. Most of these approaches are developed on a heuristic basis from an operational point of view. The use of cut–off values as in [19] or the mixture of different concepts like for instance in [3] rejects the rigorous foundations and concepts of logic programming [20]. Semantical aspects and interpretations for the truth values in the unit interval are not considered. The generalization of the resolution principle to the non–truth–functional possibilistic logic [9] can be seen as an exception in this line.

In opposition to heuristic approaches, we emphasize to develop Prolog extensions to many–valued logics on the basis of a rigorous logical framework. For two–valued logic it is sufficient to consider the set of true or provable propositions. In many–valued logic it is necessary to keep track of the truth values assigned to the propositions. Generally, instead of the exact truth value associated with a formula only lower bounds for this truth value are considered. Generalizations of classical logic to $[0,1]$–valued or fuzzy logic using this idea can be found in [24, 25, 26]. Although soundness and completeness results are obtained for these logical systems, the price for this is the addition of new axioms and inference rules involving constants for the truth values from the unit interval. This leads to the, from a practical viewpoint undesired, effect that there is no efficient proof procedure. Each proof has to be valuated according to the specified truth values and therefore the 'best' proof has to be found, which does in general not exist. Thus completeness can only be obtained by an infinite number of proof steps which can of course only be carried out theoretically. In fact, we will show that it is impossible to obtain completeness for a $[0,1]$–valued logic without the notion of infinite proofs under very general assumptions. Note that completeness is here not understood as only being able to infer all tautologies, since the propositional calculus of $[0,1]$–valued Łukasiewicz logic is complete in this sense [5, 27]. We require also to be able to derive propositions from a given set of additional axioms (a knowledge base) and to compute the corresponding truth values for the derived propositions.

In order to provide a logical system suitable for computer implementation, we consider only finitely many truth values, which does not impose severe restrictions for practical applications. In any case, no expert working with a rule based system can be expected to handle a large or even an infinite number of truth values reasonably. Besides this technical restriction we also stress to give an interpretation for the truth values. This problem is often ignored in other approaches. In order to have such an interpretation we chose n–valued Łukasiewicz logic for which a meaning for the truth values can be established on the basis of Ulam games [23]. On this basis, we are able to handle partially contradictory information.

The paper is organized as follows. Section 2 provides the formal framework for a Łukasiewicz logic based Prolog system from a purely mathematical viewpoint without giving a concrete meaning to the truth values. We obtain soundness and completeness results. Section 3 is devoted to the interpretation of the truth values as relative distance to contradiction motivated by the connection between Ulam games and Łukasiewicz logic [23]. The theoretical results of section 2 are implemented in a software tool called LULOG, which is described in section 4. Examples of the application of the software tool to an Ulam game and a typical example for rule based systems with imperfect knowledge are discussed in section 5.

2 A Language for Approximate Reasoning Based on n–Valued Łukasiewicz Logic

This section is devoted to the formal and technical background of the logical system developed in this paper from a purely mathematical point of view. A possible interpretation for the set of truth values will be discussed in the following section.

To avoid the introduction of additional axioms and inference rules, we separate the logical language from the truth values and do therefore not integrate the truth values as special logical symbols into the language as it is proposed in [24, 25, 26].

We assume L to be a first order logical language containing the logical connectives $\rightarrow$, a set $\bigoplus = \{\oplus_1, \ldots, \oplus_k\}$ of binary connectives, the unary connective $\neg$ (negation), and the universal quantifier $\forall$. Well formed formulae are defined in the usual way.

For this logical language we consider a finite set $\mathcal{T}$ of truth values. Since we will mainly concentrate on $(n+1)$–valued Łukasiewicz logic, we define without loss of generality

$$\mathcal{T} = \{0, \frac{1}{n}, \ldots, \frac{n-1}{n}, 1\}.$$

With each logical connective we associate a truth function. $\rightarrow$ is assumed to be the Łukasiewicz implication, i.e. we associate the truth function

$$\mathrm{val}_{\rightarrow} : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}, \qquad (s,t) \mapsto \min\{1 - s + t, 1\}$$

with $\rightarrow$. For the connectives in $\bigoplus$ we require the corresponding truth functions to be non–decreasing in both arguments. Typically, these connectives will be interpreted as *AND* or *OR* operators. Throughout this paper we will mainly concentrate on the following two generalizations of the logical *AND*.

$$\mathrm{val}_{\wedge} : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}, \qquad (s,t) \mapsto \min\{s,t\}$$

and

$$\mathrm{val}_{*} : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}, \qquad (s,t) \mapsto \max\{s + t - 1, 0\}.$$

The notion of a $\mathcal{T}$–valued interpretation of L is defined as in classical logic (compare for instance [20], p. 12), except that the set of truth values is $\mathcal{T}$ instead of $\{0,1\}$. The truth value assigned by a $\mathcal{T}$–valued interpretation I to a formulae φ is denoted by $/\varphi/_I$, or shortly $/\varphi/$ when it is obvious to which interpretation we refer. For the universal quantifier we define $/(\forall x)(\varphi(x))/ = \inf_x\{/\varphi(x)/\}$.

Before we can define the notion of a model, we have to generalize the concept of a set of axioms. Instead of a crisp set of axioms which is used in classical logic to characterize a theory, we consider a '$\mathcal{T}$–fuzzy set' of axioms, i.e. a mapping $a : L \rightarrow \mathcal{T}$. In classical logic, where $\mathcal{T} = \{0,1\}$, a corresponds to the characteristic function of the set of axioms. Note that if we use this characteristic function in classical logic instead of the corresponding set of axioms, the assignment of the value 0 to a formula φ by a does not mean that φ is false in the theory given by a, since it might be possible to derive φ from the axioms. Therefore, the value $a(\varphi)$ should be understood as a lower bound for the truth value of φ. This leads to the following definition of a model.

Definition 2.1 *Let $a : L \to \mathcal{T}$. A $\mathcal{T}$–valued interpretation I of L is called a model of a if $/\varphi/_I \geq a(\varphi)$ holds for all $\varphi \in L$.*

$\mathrm{Th}^{(a)} : L \to \mathcal{T}$ *denotes the infimum over all $\mathcal{T}$–valued interpretations of L that are models of a.*

We say that a implies $\varphi \in L$ to a truth degree of at least α, denoted by

$$a \models_\alpha \varphi,$$

if $\mathrm{Th}^{(a)}(\varphi) \geq \alpha$ *holds.*

Now that we have defined the semantical part of our logical system we can turn to the syntactical part by providing a proof procedure. But before we do this, we elucidate why it is impossible to define a sound finite proof method which leads to completeness with respect to the notion of a model given in definition 2.1 if we allow $\mathcal{T} = [0,1]$.

Example 2.2 Let us for the moment assume $\mathcal{T} = [0,1]$. We consider the propositional language L which contains only one propositional variable φ_1. The logical connectives in L are $\to, *$, and $\neg$ where $/\neg\varphi/ = 1 - /\varphi/$. Let us abbreviate

$$\underbrace{\varphi * \ldots * \varphi}_{m \text{ times } \varphi}$$

by φ^m. Let

$$A \;=\; \{(\neg\varphi_1) \to (\varphi_1)^m \mid m \in \mathbb{N} \text{ and } m \geq 2\}$$

and let

$$a : L \to [0,1], \qquad \varphi \mapsto \begin{cases} 1 & \text{if } \varphi \in A \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to verify that this implies $\mathrm{Th}^{(a)}(\varphi_1) = 1$. For any finite subset of $B \subseteq A$ there exists an $m_0 \in \mathbb{N}$ such that for all $m > m_0$ we have

$(\neg\varphi_1) \to (\varphi_1)^m \notin B$. It is easy to check that the interpretation induced by $/\varphi_1/ = \frac{m_0}{m_0+1}$ is a model of

$$b : L \to [0,1], \qquad \varphi \mapsto \begin{cases} 1 & \text{if } \varphi \in B \\ 0 & \text{otherwise.} \end{cases}$$

Thus $\mathrm{Th}^{(b)}(\varphi_1) \leq \frac{m_0}{m_0+1}$ holds. Any proof procedure which is allowed to carry out a finite number of steps to derive the truth value for φ from a or A, respectively, can only make use of a finite subset B of formulae in A (except if there is something like an infinite inference rule). But if this proof procedure would yield the value 1 for φ_1, it would assign this value already to φ_1 even if B is assumed to be the set of axioms. This would imply that the proof procedure is not sound.

Let us now continue with a definition of a proof procedure for our logical system with a finite set $\mathcal{T}$ of truth values. Since we are heading for a generalization of a Prolog like language and in order to obtain a simple proof procedure we will restrict ourselves from now on to Horn clause like expressions. What this means is stated in the following two definitions.

Definition 2.3 *An implication clause is a closed well formed formula of the form*

$$(\forall x_1)\dots(\forall x_k)(\varphi \to \psi) \tag{1}$$

or

$$(\forall x_1)\dots(\forall x_k)(\psi), \tag{2}$$

where ψ is an atomic formula with no other free variables than $x_1, \dots, x_k$. φ is a formula containing only connectives belonging to $\bigoplus$ and no quantifiers.

Implication clauses do not allow existential quantifiers, which is of course a restriction. We do not consider existential quantifiers here, in order to obtain a logical systems that can be used for implementation and can therefore not avoid some restrictions. In principal, the problem of ruling out existential quantifiers could be overcome by skolemization.

As 'fuzzy' Prolog programs we accept only 'fuzzy' sets of axioms $a : L \to \mathcal{T}$ that restrict to implication clauses.

Definition 2.4 *A mapping $a : L \to \mathcal{T}$ is called a $\mathcal{T}$–Ł–Prolog program if only implication clauses belong to the support of a, i.e. $a(\varphi) > 0$ implies that φ is an implication clause.*

The classical deduction schemata modus ponens and substitution can be easily incorporated into our logical system. Of course, we have to keep track of the corresponding lower bounds for the truth values. In part (i) of the following definition the calculation of greatest lower bound for the corresponding truth values based on the truth function $\to$ is incorporated into the deduction schemata modus ponens and substitution.

Definition 2.5 *Let $a, b : L \to \mathcal{T}$ be a $\mathcal{T}$–Ł–Prolog program.*

(i) b is directly derivable from a if

(a) there exists an implication clause $(\forall x_1)\ldots(\forall x_k)(\varphi \to \psi)$ in L such that

(a1) If ψ_0 is an implication clause with free variables $x_1, \ldots, x_r$ and $\psi_0 \neq \psi$, then $a((\forall x_1)\ldots(\forall x_r)(\psi_0)) = b((\forall x_1)\ldots(\forall x_r)(\psi_0))$.

(a2)

$$b((\forall x_1)\ldots(\forall x_k)(\psi)) = \max\{ \begin{array}{l} /(\forall x_1)\ldots(\forall x_k)(\varphi)/_a \\ +a((\forall x_1)\ldots(\forall x_k)(\varphi \to \psi)) \\ -1, \\ a((\forall x_1)\ldots(\forall x_k)(\psi)) \ \}, \end{array} \quad (3)$$

where the value $/(\forall x_1)\ldots(\forall x_k)(\varphi)/_a$ is obtained by considering the Herbrand universe of L and valuating atomic formulae according to a,

holds or

(b) there exists an implication clause $(\forall x_1)\ldots(\forall x_k)(\chi)$ and terms $t_{i_1}, \ldots, t_{i_r}$ ($i_1, \ldots i_r \in \{1, \ldots, k\}$) without free variables such that for the formula χ', which is obtained by substituting x_{i_j} ($j = 1, \ldots, r$) by t_{i_j} in χ and quantifying over the remaining free variables,

$$b(\chi') = \max\{a((\forall x_1)\ldots(\forall x_k)(\chi)), a(\chi')\}. \quad (4)$$

is satisfied.

(ii) *b is derivable from a, abbreviated by $a \lhd b$, if there is a sequence $a_0, \dots, a_n : L \to \mathcal{T}$ of $\mathcal{T}$–L–Prolog programs where a_{k+1} is directly derivable from a_k for each $k \in \{0, \dots, n-1\}$ and $b \le a_n$ holds.*

(iii) *The mapping $\mathrm{th}^{(a)} : L \to \mathcal{T}$ is given by*

$$\mathrm{th}^{(a)}(\varphi) = \begin{cases} \sup\{b(\varphi) \mid a \lhd b\} & \text{if } \varphi \text{ is an implication clause} \\ & \text{of the form (2)} \\ 0 & \text{otherwise.} \end{cases}$$

(iv) *An implication clause $\varphi \in L$ of the form of the form (2) is derivable from a with a truth degree of at least $\alpha \in \mathcal{T}$, abbreviated by*

$$a \vdash_\alpha \varphi,$$

if $\mathrm{th}^{(a)}(\varphi) \ge \alpha$ holds.

The application of modus ponens and the substitution of free variables by other terms is formalized in (i)(a) and (i)(b), respectively, in the above definition. (ii) describes the application of a finite number of deduction steps of the form introduced in (i). Since we assume $\mathcal{T}$ to be the finite chain $\{0, 1/n, \dots, (n-1)/n, 1\}$ the supremum in (iii) is in fact a maximum. Thus, for all implication clauses φ of the form (2), there exists a $\mathcal{T}$–L–Prolog program b_φ derivable from a such that $b_\varphi(\varphi) = \mathrm{th}^{(a)}(\varphi)$.

Example 2.6 Consider the following three logical formulae and a $\mathcal{T}$–L–Prolog program a, where $\mathcal{T} = \{0, 0.1, \dots, 0.9, 1\}$, that assigns the indicated values to these formulae.

$(\forall x)(P(x) \wedge Q(x) \to R(x))$	0.9
$P(X_0)$	0.8
$Q(X_0)$	0.7

where P, Q, and R are symbols for predicates and X_0 is an individual constant. To all other formulae φ the value zero is assigned by a. Applying part (i)(b) of Definition 2.5 to first formula, we can derive

$$P(X_0) \wedge Q(X_0) \to R(X_0) \qquad 0.9. \tag{5}$$

Technically speaking, we can derive from a the $\mathcal{T}$–L–Prolog program b that assigns the same values to formulae as a, except for the formula $P(X_0) \wedge Q(X_0) \rightarrow R(X_0)$ to which b assigns the value 0.9.

Assuming that $/\varphi \wedge \psi/ = \max\{/\varphi/ + /\psi/ - 1, 0\}$ is the truth function associated with the conjunction $\wedge$, part (i)(a2) of Definition 2.5 applied to equation (5) yields

$$R(X_0) \qquad 0.4 = \max\{\max\{0.8 + 0.7 - 1, 0\} + 0.9 - 1, 0\}$$

when we take the values assigned to $P(X_0)$ and $Q(X_0)$ into account. Thus we have shown $\mathrm{th}^{(a)}(R(X_0)) \geq 0.4$. In fact, we have even $\mathrm{th}^{(a)}(R(X_0)) = 0.4$, since the proof is the one which gives the greatest value for $R(X_0)$. We may also write

$$a \vdash_\alpha R(X_0)$$

for any $\alpha \in \{0, 0.1, \ldots, 0.4\}$.

Theorem 2.7 *Let $a : L \rightarrow \mathcal{T}$ be a $\mathcal{T}$–*L*–Prolog program and let φ be an implication clause of the form* (2). *Then*

$$\mathrm{th}^{(a)}(\varphi) \leq \mathrm{Th}^{(a)}(\varphi)$$

holds.

Proof. We have to prove that , if I is a model of a $\mathcal{T}$–L–Prolog program then I is also a model for any $\mathcal{T}$–L–Prolog program directly derivable from the given $\mathcal{T}$–L–Prolog program. Let I be a model of a and let b be directly derivable from a.

Case 1. b is obtained from a by applying (3).

Since I is a model of a and according to the monotonicity of the truth functions in $\bigoplus$, we derive

$$\begin{array}{rcll} /(\forall x_1)\ldots(\forall x_k)(\psi)/ & \geq & \max\{ & /(\forall x_1)\ldots(\forall x_k)(\varphi)/ \\ & & & +/(\forall x_1)\ldots(\forall x_k)(\varphi \rightarrow \psi)/_a - 1, \\ & & & /(\forall x_1)\ldots(\forall x_k)(\psi)/\ \} \\ & \geq & \max\{ & /(\forall x_1)\ldots(\forall x_k)(\varphi)/_a \\ & & & +/(\forall x_1)\ldots(\forall x_k)(\varphi \rightarrow \psi)/_a - 1, \\ & & & /(\forall x_1)\ldots(\forall x_k)(\psi)/_a\ \} \\ & = & b((\forall x_1)\ldots(\forall x_k)(\psi)). & \end{array}$$

Case 2. b is obtained from a by applying (4).
For the same reasons as in case 1, we obtain

$$\begin{aligned} /\chi'/ &= \max\{/(\forall x_1)\dots(\forall x_k)(\chi)/, /\chi'/\} \\ &\geq \max\{a((\forall x_1)\dots(\forall x_k)(\chi)), a(\chi')\}. \end{aligned}$$

$\square$

Corollary 2.8 (Soundness) *Let $a : L \to \mathcal{T}$ be a $\mathcal{T}$-Ł-Prolog program, let φ be an implication clause of the form* (2)*, and let $\alpha \in \mathcal{T}$. Then*

$$a \vdash_\alpha \varphi \quad \Longrightarrow \quad a \models_\alpha \varphi$$

holds.

Theorem 2.9 *Let $a : L \to \mathcal{T}$ be a $\mathcal{T}$-Ł-Prolog program and let φ be an implication clause of the form* (2)*. Then*

$$\mathrm{th}^{(a)}(\varphi) \geq \mathrm{Th}^{(a)}(\varphi).$$

holds.

Proof. Let U be the Herbrand universe of L. We show that the Herbrand interpretation I induced by $\mathrm{th}^{(a)}$ is compatible with a. For implication clauses χ of the form (2) the definition of $\mathrm{th}^{(a)}$ yields

$$/\chi/ \geq a(\chi).$$

Thus, we only have to consider implication clauses like $(\forall x_1)\dots(\forall x_k)(\varphi \to \psi)$ of the form (1) where

$$/(\forall x_1)\dots(\forall x_k)(\varphi \to \psi)/ < a((\forall x_1)\dots(\forall x_k)(\varphi \to \psi)).$$

There exists a tuple $u = (u_1, \dots, u_k) \in U^k$ such that

$$/\varphi(u) \to \psi(u)/ < a((\forall x_1)\dots(\forall x_k)(\varphi \to \psi)) \tag{6}$$

holds where $\varphi(u)$ and $\psi(u)$ are obtained by substitution of $x_1, \dots, x_k$ by $u_1, \dots, u_k$ in φ and ψ, respectively. By applying part (i)(b) of Definition 2.5

to the formula $(\forall x_1)\dots(\forall x_k)(\varphi \to \psi)$ by substituting $x_1,\dots,x_k$ by $u_1,\dots,u_k$ we derive $b : L \to \mathcal{T}$ directly from a. Then we have

$$\begin{aligned} 1 &\geq b(\varphi(u) \to \psi(u)) \\ &\geq a((\forall x_1)\dots(\forall x_k)(\varphi \to \psi)) \\ &> /\varphi(u) \to \psi(u)/ \\ &= \min\{1 - /\varphi(u)/ + /\psi(u)/, 1\} \\ &= 1 - /\varphi(u)/ + /\psi(u)/. \end{aligned} \tag{7}$$

(7) implies

$$/\varphi(u)/ + b(\varphi(u) \to \psi(u)) - 1 > /\psi(u)/.$$

According to the remark after definition 2.5, there is a $\mathcal{T}$–L–Prolog program a' derivable from a such that $/\varphi(u)/ = a'(\varphi(u))$. Obviously, the $\mathcal{T}$–L–Prolog program $b' = \max\{a', b\}$ is also derivable from a. For b' we have

$$b'(\varphi(u)) + b'(\varphi(u) \to \psi(u)) - 1 > /\psi(u)/.$$

From b' we can directly derive the $\mathcal{T}$–L–Prolog program $\tilde{b}$ with

$$\tilde{b}(\psi(u)) = b'(\varphi(u)) + b'(\varphi(u) \to \psi(u)) - 1 > /\psi(u)/.$$

By definition $\tilde{b}$ is also derivable from a. But this leads to the contradiction

$$/\psi(u)/ = \mathrm{th}^{(a)}(\psi(u)) \geq \tilde{b}(\psi(u)) > /\psi(u)/.$$

$\square$

Corollary 2.10 (Completeness) *Let $a : L \to \mathcal{T}$ be a $\mathcal{T}$–*L*–Prolog program, let φ be an implication clause of the form* (2)*, and let $\alpha \in \mathcal{T}$. Then*

$$a \models_\alpha \varphi \quad \Longrightarrow \quad a \vdash_\alpha \varphi$$

holds.

Corollaries 2.8 and 2.10, respectively Theorems 2.7 and 2.9, guarantee soundness and completeness for the proof procedure described in Definition 2.5 with respect to the semantics of definition 2.1. For $\mathcal{T}$–L–Prolog program it is obviously sufficient to restrict to implication clauses of the form (2) for soundness and completeness. How we can make use of these results for an implementation is shown in section 4.

The same soundness and completeness results can be obtained for the infinite valued logic with $\mathcal{T} = [0,1]$ [14] for the price that the value $\mathrm{th}^{(a)}(\varphi)$ cannot be obtained by a finite number of direct derivations. In [14] also the Gödel implication is considered as a possible truth function for the implication. A Prolog system based on this implication and the set of truth values $\mathcal{T} = [0,1]$ can be shown to be equivalent to Prolog based on possibilistic logic [16]. For such $[0,1]$–valued logical system a probabilistic semantics can be provided and the use of the Lukasiewicz implication then corresponds to a possibly overcautious application of deduction [15].

3 Interpretation of the Truth Values

The previous section approached the definition of a Lukasiewicz logic based Prolog from a purely formal point of view by generalizing concepts from classical logic to many–valued logic. The truth values are considered as abstract symbols and no interpretation for intermediate truth values between 0 and 1 is provided. In this section we address this problem of interpreting truth values, for which pick up Mundici' idea to consider truth values in Lukasiewicz logic as distance from contradiction motivated by Ulam games. Let us briefly recall Mundici's argumentation [23].

Mundici considers an Ulam game [30], which is a simple 'guess a number from the set $S = \{1,\ldots,p\}$ by asking questions to be answered by yes or no', except that the person who provides the answers to the questions is allowed to lie (at most) ℓ times. In section 5 an example of an Ulam game is discussed.

Let us assume that the questions $Q_1,\ldots,Q_q$ were asked and the given answers where $A_1,\ldots,A_q$. We fix a number $s \in S$ from the set of possible numbers for the moment and try to represent our knowledge induced by the given answers with respect to this number s. If $\mathrm{false}(s)$ denotes the number of lies in the answers $A_1,\ldots,A_q$, given s would be the number to be guessed, then we can represent our state of knowledge by the rational number

$$k_{A_1,\ldots,A_q}(s) \;=\; 1 - \frac{\min\{\mathrm{false}(s), \ell+1\}}{\ell+1} \quad \in \left\{0, \frac{1}{\ell+1}, \ldots, \frac{\ell}{\ell+1}, 1\right\}.$$

$k_{A_1,\ldots,A_q}(s) = d/(\ell+1)$ means that $(\ell+1-d)$, or in the case of $d=0$ more than ℓ, lies are among the answers $A_1,\ldots,A_q$. Thus $k_{A_1,\ldots,A_q}(s)$ can be viewed as the relative distance, measured in units $(1+\ell)$, from contradiction, i.e. from falsifying too many answers assumed s is the number to be guessed.

If we consider only a single answer A_t, there are two possibilities for the value $k_{A_t}(s)$. In the case that s falsifies A_t, we obtain $k_{A_t}(s) = \ell/(\ell+1)$, otherwise we have $k_{A_t}(s) = 1$. Now it easy to check that $k_{A_1,A_2}(s) = \max\{k_{A_1}(s) + k_{A_2}(s) - 1, 0\}$ holds, or more generally

$$k_{A_1,\ldots,A_q}(s) \;=\; \max\{k_{A_1,\ldots,A_t}(s) + k_{A_{t+1},\ldots,A_q}(s) - 1, 0\}. \qquad (8)$$

Equation (8) shows that our state of knowledge can be computed by the Lukasiewicz conjunction

$$/\varphi * \psi/ = \max\{/\varphi/ + /\psi/ - 1, 0\}$$

in an $(\ell+2)$–valued logic. Note that this implies $/\varphi * \varphi/ < /\varphi/$ except for $/\varphi/ \in \{0,1\}$. This phenomenon is according to the effect that in an Ulam game with lies the repetition of a question leads to more information than the same question asked only once.

Of course, we have not developed the Lukasiewicz logic based Prolog just for the sake of Ulam games. The above given interpretation from Mundici for Lukasiewicz logic should be understood as a motivation to a possible interpretation of truth values in terms of relative distance to contradiction. Abstracting from Ulam games we can interpret the truth values in the following way. The person who specifies the truth values has an understanding of the 'real world' which contains partial contradictions or incoherencies. Due to this incoherence the real world is associated with an inconsistent set of (not necessarily explicitly specified) statements. These statements play the role of the questions and answers in the Ulam game. It is assumed that, although the statements might be contradictory, a maximum of ℓ of these statements can be falsified. The person then evaluates each logical proposition φ with respect to the statements about the world and assigns, as in the Ulam game, the truth value $d/(\ell+1)$ if φ falsifies $(\ell+1-d)$ of the statements.

This yields immediately an interpretation of the Lukasiewicz conjunction $*$ as a logical *and*, since $/\varphi * \psi/ = \max\{/\varphi/ + /\psi/ - 1, 0\}$ is the distance of the proposition (φ *and* ψ) in the pessimistic case that the sets of statements about the real world falsified by φ and ψ are disjoint, at least if $/\varphi * \psi/ > 0$.

The optimistic case, i.e. the set of statements falsified by φ is contained in the set of statements falsified by ψ or vice versa, would lead to the minimum as the truth function for the logical *and*.

In order to find a truth function for the implication, we have to explain how we treat propositions with an implication. Since we are interested in the truth value as the relative distance from contradiction, we think of implication in the following way. In classical logic the validity of the implication $\varphi \rightarrow \psi$ allows us to assume ψ instead of φ without making an additional mistake, i.e. if we would have to bet on φ or ψ, we would be better off to bet on ψ. In the same way, a truth value of 1 for the implication $\varphi \rightarrow \psi$ in our interpretation of truth values should guarantee that ψ does not falsify more statements than φ. If ψ does falsify more statements than φ the implication is 'closer' to contradiction and its truth value should tell us how near it is to contradiction. This leads to the Łukasiewicz implication as the truth function for $\rightarrow$, i.e. if φ and ψ falsify m_φ and m_ψ statements, respectively, and if we assume that $m_\varphi \leq m_\psi \leq \ell$, then we assign the truth value

$$\frac{\ell + 1 - (m_\psi - m_\varphi)}{\ell + 1}$$

to the implication $\varphi \rightarrow \psi$.

It is of course possible to motivate other logical connectives in this framework for handling distance from contradiction. But for our purposes the above mentioned logical operators *and* as Łukasiewicz conjunction or minimum, and implication as Łukasiewicz implication are sufficient.

It should be emphasized that the above mentioned interpretation of the truth values. There are of course others which are intended to model different phenomena. Various examples of applications of many–valued logics in expert systems can be found in [1].

4 LULOG – An Implementation

The theoretical results provided in section 2 lead to an implementation of a software tool which we call LULOG. LULOG is an interpreter for a Łukasiewicz logic based Prolog. LULOG itself is written in CommonLISP. The basic syntax of LULOG in a Backus–Naur–like form is shown in table

LULOG expression	::= query | entry
query	::= expression | connective expression
entry	::= (, `ASSERT!`, {data | rule},)
rule	::= (, `RULE`, expression, {expression | connective expression}, truth value
connective expression	::= (, {`AND` | `AND*`}, {expression | connective expression}, {expression | connective expression}*,)
truth value	::= 0 | 1 |… n
data	::= a LISP list (of symbols for predicates and individuals) whose last element is a truth value
expression	::= a LISP list (of symbols for predicates and individuals)

Table 1: The syntax of LULOG.

1. Since '(' and ')' are symbols in the LULOG language, we have used { and } for grouping expressions, deviating from the standard Backus–Naur form.

In LULOG the set of truth values $\{0, 1/n, \ldots, (n-1)/n, 1\}$ used in section 2 is replaced by the set $\{0, 1, \ldots, (n-1), n\}$ so that only integer arithmetic is needed in the program leading to a faster execution.

Examples for entries in the knowledge base are

```
(ASSERT! (ANIMAL TWEETY 5))
(ASSERT! (HAS-FEATHERS TWEETY 6))
(ASSERT! (RULE (BIRD ?X) (AND (ANIMAL ?X) (HAS-FEATHERS
?X)) 4)).
```

The last rule states that for any x, we can derive that x is a bird if x is an animal and has feathers (taking the corresponding truth values into account). Variables are marked by a question mark as the first letter.

Typical queries are

```
(HAS-FEATHERS TWEETY)
(AND* (ANIMAL ?X) (HAS-FEATHERS ?X)).
(?WHAT TWEETY)
```

The answer to the first query yields the truth value with which the predicate *has–feathers* holds for *tweety*, whereas the second query lists all individuals, to which the two predicates *animal* and *has–feathers* apply, and computes for each individual the Lukasiewicz conjunction of the corresponding truth values. The last query yields all predicates with the corresponding truth values which can be applied to *tweety*. More examples for entries in the knowledge base and queries can be found in section 5.

The basic inference mechanism of LULOG is modus ponens. Besides modus ponens substitution of variables is necessary. A proof for a query is found by backward chaining, i.e. if we want to prove $\varphi(x, y, \ldots, z)$, we look for facts containing φ, yielding directly a truth value, or for rules with φ in the head, and then unify the body of the rule correspondingly and continue with the unified predicates in the body of the rule in the same way until we reach a fact again. If a proof is found by this procedure, the backward chaining has to be retraced in the opposite direction in order to compute the corresponding truth value. Note that we have to find the proof yielding the maximal truth value, so that we can stop the search procedure only, when we have found a proof that gives the maximal truth value n appearing in the set of truth values, or when all proofs are examined.

If it is not necessary to find the greatest derivable truth value for a query, one may also interrupt the program LULOG in order to get the value of the best proof found until the time of interruption. In order to increase the chances for finding good proofs in an early state of the search, LULOG chooses the rule to which the highest truth value is assigned, when there is more than one possible rule for the backward chaining procedure.

5 Application Examples

In this section we discuss the application of our Lukasiewicz logic based Prolog system to two examples. For those readers who are familiar with the programming languages Prolog and Lisp, we have also written down parts of the dialog with the machine.

Example 5.1 The first example is very simple and does not involve any chained inference. We try to solve an Ulam game with $S = \{1, 2, 3, 4, 5\}$ as the set from which the number to be guessed is chosen and a permission

Question		Answer
Q_1	The number to be guessed is a prime.	no
Q_2	The number to be guessed is even.	yes
Q_3	The number to be guessed is even.	no
Q_4	The number to be guessed is 1 or 5.	no
Q_5	The number to be guessed is a square.	yes
Q_6	The number to be guessed is less than 4.	no

Table 2: An Ulam game.

of a maximum of two lies. Therefore, as explained in section 3, we have to choose a 4–valued logic. As mentioned in section 4, we take the set $\{0, 1, 2, 3\}$ as truth values instead of $\{0, 1/3, 2/3, 1\}$. The interpretation of these truth values is

0: more than two lies

1: two lies

2: one lie

3: no lies.

Table 2 shows a protocol of an Ulam game. In order to make use of our Lukasiewicz logic based Prolog system, we introduce six predicates *answer_1*, ..., *answer_6*. Starting from an empty knowledge base, we add for each $i \in \{1, \ldots, 6\}$ and each $s \in S = \{1, \ldots, 5\}$ the fact *answer_i*(s) with truth value 2 (lie) if the number s would make the answer to question Q_i a lie, and with truth value 3 (no lie) if the answer to question Q_i is correct, assumed that s is the number to be guessed. As an example the corresponding entries in the knowledge base for Q_1 and Q_5 are shown in table 3.

In order to see, how many lies are under the answers for each $s \in \{1, \ldots, 5\}$, we simply have to compute the Lukasiewicz conjunction for each s. This is done automatically by the Lukasiewicz logic based Prolog when we ask for the truth value of the Lukasiewicz conjunction of the predicates *answer_i* for a non–specified variable. The truth value 3,2,1, and 0 correspond to no lie, one lie, two lies, more than three lies. Table 4 shows the

s	Q_1	Q_5
1	`(ASSERT! (ANSWER1 ONE 3))`	`(ASSERT! (ANSWER5 ONE 3))`
2	`(ASSERT! (ANSWER1 TWO 2))`	`(ASSERT! (ANSWER5 TWO 2))`
3	`(ASSERT! (ANSWER1 THREE 2))`	`(ASSERT! (ANSWER5 THREE 2))`
4	`(ASSERT! (ANSWER1 FOUR 3))`	`(ASSERT! (ANSWER5 FOUR 3))`
5	`(ASSERT! (ANSWER1 FIVE 3))`	`(ASSERT! (ANSWER5 FIVE 2))`

Table 3: Entries in the knowledge base for Q_1 and Q_5.

`?X =1= FIVE`	`?X =0= FIVE`
`?X =2= FOUR`	`?X =2= FOUR`
`?X =1= THREE`	`?X =0= THREE`
`?X =1= TWO`	`?X =0= TWO`
`?X =2= ONE`	`?X =0= ONE`

Table 4: The state of knowledge after the first 3 (left) and 6 (right) answers.

results after the first three answers and after all six answers, respectively. The corresponding results are obtained by the LULOG commands

- `(AND* (ANSWER1 ?X) (ANSWER2 ?X) (ANSWER3 ?X))` and
- `(AND* (ANSWER1 ?X) (ANSWER ?X) (ANSWER3 ?X)`
  `(ANSWER4 ?X) (ANSWER5 ?X) (ANSWER6 ?X))`,

respectively.

From table 4 we can see that after the first three answers still all numbers are possible, but after six answers we know that the correct guess must be 4 (corresponding to one lie only).

Example 5.2 In this example we discuss a shortened version of the test–example examined in [29]. The following statements have to be integrated into the knowledge base.

(s1) Students are young.

(s2) Young people are single.

(s3) Students who have children are married or cohabitants.

(s4) Cohabitants are young.

(s5) Single, married, and cohabitant are mutually exclusive.

In order to describe these statements in a formal language, the unary predicates $sdnt(x)$, $yng(x)$, $sng(x)$, $pnt(x)$, $mrd(x)$, and $chbt(x)$ are introduced, meaning that x is respectively student, young, single, parent (having one or more children), married, and cohabitant. In addition we use the proposition *cntr* for representing a contradiction.

Only (s5) is understood as a proposition in classical logic, the other statements are true in most, but not in all cases. The statements, for instance (s1), can be written in Prolog–like style in the form

$$yng(x) \leftarrow sdnt(x).$$

Since we cannot be sure that (s1) always holds, we have to assign an appropriate truth value to this rule, indicating the relative distance from contradiction. We choose a 7–valued logic with the truth values $0, \ldots, 6$. The truth values assigned to the statements (s1), (s2), and (s4) are 5, 4, and 4, respectively. (s3) cannot be represented directly in the language LULOG, since in the head of a rule no conjunctions or disjunctions are admitted. In this case, we can make use of the fact that married and cohabitant are mutually exclusive, splitting (s3) into the two rules

(s3a) Students who have children are married.

(s3b) Students who have children are cohabitants.

We assign to each rule the truth value 3 so that the Lukasiewicz conjunction of 'married' and 'cohabitant' obtained from these rules, when we are dealing with a student with children, always yields the value 0.

(s5) is expressed by the following three rules, again written in Prolog–like style, which are always valid and are therefore assigned the truth value 6.

	Entry in the knowledge base
(s1)	`(ASSERT! (RULE (YNG ?X) (SDTN ?X) 5))`
(s2)	`(ASSERT! (RULE (SNG ?X) (YNG ?X) 4))`
(s3a)	`(ASSERT! (RULE (MRD ?X) (AND (SDTN ?X) (PNT ?X)) 3))`
(s3b)	`(ASSERT! (RULE (CHBT ?X) (AND (SDTN ?X) (PNT ?X)) 3))`
(s4)	`(ASSERT! (RULE (YNG ?X) (CHBT ?X) 4))`
(s5a)	`(ASSERT! (RULE (CNTR) (AND* (SNG ?X) (MRD ?X)) 6))`
(s5b)	`(ASSERT! (RULE (CNTR) (AND* (SNG ?X) (CHBT ?X)) 6))`
(s5c)	`(ASSERT! (RULE (CNTR) (AND* (MRD ?X) (CHBT ?X)) 6))`

Table 5: The knowledge base for example 5.2.

(s5a) $\mathit{cntr} \leftarrow \mathit{sng}(x) * \mathit{mrd}(x)$

(s5b) $\mathit{cntr} \leftarrow \mathit{sng}(x) * \mathit{chbt}(x)$

(s5c) $\mathit{cntr} \leftarrow \mathit{mrd}(x) * \mathit{chbt}(x)$

The use of the Lukasiewicz conjunction for (s5a)–(s5c) admits the possibility that a person might be a assigned non–zero truth values for all the three predicates *sng*, *mrd*, and *chbt*, but still having the truth value 0 for *cntr*. This holds as long as the sum of the truth values of two of these predicates is always less than 6. It means that we are accepting a partial contradiction.

The complete knowledge base in LULOG is shown in table 5.

Now let us assume that we know that Lea is a student and that she is possibly a mother, since we saw her with a child resembling her face on the campus. So we add to the knowledge the two facts *sdnt*(lea) and *pnt*(lea) with truth value 6 and 5, respectively. Asking the question who is single, we obtain the answer that *sng*(lea) is satisfied with truth value 3 (and no more individuals are listed, since *sng* cannot be applied to someone else included in the knowledge base at the moment). Asking for cohabitants or married persons we get that *chbt*(lea) or *mrd*(lea) both hold with truth value 2. Although single, cohabitant, and married are mutually exclusive, we obtain that *cntr* (contradiction) holds with truth value 0. This is due to the low truth value for *chbt* and *mrd*, so that we still can accept this partial contradiction. Table 6 shows a LULOG protocol of the corresponding questions and answers.

Additional entries for the knowledge base	
`(ASSERT! (SDNT LEA 6))`	
`(ASSERT! (PNT LEA 5))`	
Question	Answer
`(SNG ?X)`	`?X =3= LEA`
`(MRD ?X)`	`?X =2= LEA`
`(CHBT ?X)`	`?X =2= LEA`
`(CNTR)`	`Valid with value 0`

Table 6: The LULOG dialogue concerning Lea.

Additional entries for the knowledge base	
`(ASSERT! (SDNT PAUL 6))`	
`(ASSERT! (CHBT PAUL 6))`	
`(ASSERT! (CHBT LEA 6))`	
Question	Answer
`(SNG ?X)`	`?X =3= LEA`
	`?X =3= PAUL`
`(CNTR)`	`Valid with value 3`

Table 7: The LULOG dialogue concerning Lea and Paul.

Let us finally assume that we learn that Paul who is also a student is Lea's cohabitant. So we add the facts *sdtn*(paul), *chbt*(paul), and *chbt*(lea) with truth value 6 to the knowledge base. When we now ask for single people we obtain that the predicate *sng* holds with truth value 3 for Lea as well as for Paul. Together with the knowledge that Lea and Paul are cohabitants, this leads to a value of 3 for *cntr* (contradiction). The protocol of the corresponding LULOG dialogue is shown in table 7.

6 Conclusions

We have provided a formal framework for a Łukasiewicz logic based Prolog and have discussed a software tool which was developed on this theoretical basis. In principal it is possible to drop the restriction to a finite set of truth values, especially since in practice we always have to deal with a finite knowledge base. But the introduction of real numbers from the unit interval as truth values would enforce us to give up the simple integer operations for the finite–valued case. This is a severe disadvantage since, in opposition to ordinary Prolog, we have to look for the proof yielding the greatest truth value. This means that we have to search through the whole proof tree in all cases, which is of course much more complex than breaking up after the first proof is found as it is done in Prolog.

We have used the Łukasiewicz implication for semantical reasons, in order to be able to apply the interpretation of the truth values of Łukasiewicz logic introduced by Mundici [23]. The theoretical results of section 2 are still valid for other implication operators like the Gödel implication. Of course, equation (3) in definition 2.5 has to be modified accordingly. To adapt the modification for the software tool LULOG, only this formula used in the valuation of proofs has to be redefined.

As ordinary Prolog the LULOG system is only based on a restricted subset of first order logic, mainly by avoiding negation. An extension to the full calculus of first order Łukasiewicz logic would lead to the same complexity and undecidability problems that appear already for classical logic, since classical logic is included in LULOG simply by choosing $\{0, 1\}$ as the set of truth values. And in contrast to classical logic, we are in the bad situation that first order Łukasiewicz logic is not complete [28].

Nevertheless, LULOG can cope with various problems connected to reasoning with partially inconsistent knowledge. In many other fuzzy Prolog systems often only the proof procedure of Prolog is fuzzified so that it is impossible to describe suitable semantics for such systems. LULOG is based on a pure logical approach where syntax and semantics are strictly separated, guaranteeing for a clear interpretation of the system.

References

[1] B.M. Ayyub, M.M. Gupta, L.N. Kanal (eds.), Analysis and Management of Uncertainty: Theory and Applications. North–Holland, Amsterdam (1992).

[2] F. Bacchus, Representing and Reasoning with Probabilistic Knowledge. MIT Press, Cambridge, Massachusetts (1990).

[3] J.F. Baldwin, Fast Operations on Fuzzy Sets in the Abstract FRIL Machine. Proc. IEEE Intern. Conf. on Fuzzy Systems, San Diego (1992), 803–809.

[4] P. Besnard, Default Logic. Springer–Verlag, Berlin (1989).

[5] C.C. Chang, Algebraic Analysis of Many–Valued Logics. Trans. American Mathematical Society 88 (1958), 467–490.

[6] W.F. Clocksin, C.S. Mellish, Programming in Logic (2nd ed.). Springer–Verlag, Berlin (1984).

[7] R. Cordes, R. Kruse, H. Langendörfer, H. Rust, Prolog (3rd ed.) (in German). Vieweg, Braunschweig (1992).

[8] D. Dubois, J. Lang, H. Prade, Fuzzy Sets in Approximate Reasoning, Part 2: Logical Approaches. Fuzzy Sets and Systems 40 (1991), 203–244.

[9] D. Dubois, H. Prade, Resolution Principles in Possibilistic Logic. Intern. Journ. Approximate Reasoning 4 (1990), 1–21.

[10] D. Dubois, H. Prade, Epistemic Entrenchment and Possibilistic Logic. Artificial Intelligence 50 (1991), 223-239.

[11] D. Dubois, H. Prade, Fuzzy Sets in Approximate Reasoning, Part 1: Inference with Possibility Distributions. Fuzzy Sets and Systems 40 (1991), 143–202.

[12] S. Gottwald, Fuzzy Sets and Fuzzy Logic. Vieweg, Wiesbaden (1993).

[13] M. Ishizuka, N. Kaisai, Prolog–ELF Incorporating Fuzzy Logic. Proc. 9th IJCAI, Los Angeles (1985), 701–703.

[14] F. Klawonn, Prolog Extensions to Many–Valued Logics. In: U. Höhle, E.P. Klement (eds.), Proc. 14th Linz Seminar on Fuzzy Set Theory: Non–Classical Logics and their Applications. Johannes Kepler Universität, Linz (1992), 42–45.

[15] F. Klawonn, J. Gebhardt, R. Kruse, Logical Approaches to Uncertainty and Vagueness in the View of the Context Model. Proc. IEEE International Conference on Fuzzy Systems 1992, IEEE, San Diego (1992), 1375–1382.

[16] F. Klawonn, J. Gebhardt, R. Kruse, The Context Model from the Viewpoint of Logic. In: K.–W. Hansmann, A. Bachem, M. Jarke, W.E. Katzenberger, A. Marusev, Operations Research Proceedings 1992. Springer–Verlag, Berlin (1993), 288–295.

[17] R. Kruse, J. Gebhardt, F. Klawonn, Foundations of Fuzzy Systems. Wiley, Chichester (1994).

[18] R. Kruse, E. Schwecke, J. Heinsohn, Uncertainty and Vagueness in Knowledge Based Systems: Numerical Methods. Springer–Verlag, Berlin (1991).

[19] R.C.T. Lee, Fuzzy Logic and the Resolution Principle. Journ. of the Association for Computing Machinery 19 (1972), 109–119.

[20] J.W. Lloyd, Foundations of Logic Programming (2nd ed.). Springer–Verlag, Berlin (1987).

[21] T.P. Martin, J.F. Baldwin, B.W. Pilsworth, The Implementation of FPROLOG – A Fuzzy Prolog Interpreter. Fuzzy Sets and Systems 23 (1987), 119–129.

[22] M. Mukaidono, Z.L. Shen, L. Ding, Fundamentals of Fuzzy Prolog. Intern. Journ. Approximate Reasoning 3 (1989), 179–193.

[23] D. Mundici, Ulam Games, Lukasiewicz Logic, and AF $C^\star$–Algebras. Fundamenta Informaticae 18 (1993), 151–161.

[24] V. Novák, On the Syntactica–Semantical Completeness of First Order Fuzzy Logic, Part I: Syntax and Semantics. Kybernetica 26 (1990), 47–66.

[25] V. Novák, On the Syntactica–Semantical Completeness of First Order Fuzzy Logic, Part II: Main Results. Kybernetica 26 (1990), 134–154.

[26] J. Pavelka, On Fuzzy Logic I, II, III. Zeitschr. Math. Logik Grundl. Math. 25 (1979), 45–52, 119–134, 447–464.

[27] A. Rose, J.B. Rosser, Fragments of Many–Valued Statement Calculi. Trans. American Mathematical Society 87 (1958), 1–53.

[28] B. Scarpellini, Die Nichtaxiomatisierbarkeit des unendlichwertigen Prädikatenkalküls von Łukasiewicz. Journal of Symbolic Logic 27 (1962), 159–170.

[29] L. Sombé, Reasoning under Incomplete Information in Artificial Intelligence. Wiley, New York (1990).

[30] S.M. Ulam, Adventures of a Mathematician. Scribner's, New York (1976).

[31] M. Umano, Fuzzy Set Prolog. Proc. 2nd IFSA Congress, Tokyo (1987), 750–753.