

# A class of theorems in Łukasiewicz logic for benchmarking automated theorem provers

Robert Rothenberg

School of Computer Science  
University of St Andrews  
St Andrews, Fife KY16 9SX  
Scotland, UK  
rr@cs.st-andrews.ac.uk

**Abstract.** We outline semantic proofs of two related classes theorems of Łukasiewicz logic, one of which is more difficult to prove than the other in the calculi that we have surveyed. We discuss some reasons why, and present benchmarks for some implementations of these proof systems against these theorems.

## 1 Motivation

When developing experimental implementations of theorem provers based on symbolic calculi for Łukasiewicz logic, we noted that one direction ( $\Rightarrow$ ) of the equivalence<sup>1</sup>

$$(A \wedge B) \oplus (A \wedge B) \equiv (A \oplus A) \wedge (B \oplus B)$$

was easily proven, while the converse ( $\Leftarrow$ ) was not easily proven— the programs were run for long periods of time (sometimes days) before returning with a negative answer (possibly due to flaws in the implementation) or terminating due to stack overflows.

Further investigation led to the proof of a class of equalities for which proof search in one direction appears significantly harder than proof search in the opposite direction. Why this is so, and whether the complexity of proof search for the hard cases can be reduced, is a matter of interest.

We add that with the development of several calculi and tableaux methods for proving theorems of Łukasiewicz logic in recent years (e.g. [13, 18, 6, 17]), it will be useful to have formulae suitable for comparing these proof systems and their implementations.

## 2 Łukasiewicz Logic

Łukasiewicz logics are a class of many-valued logics originally introduced in the 1920s [15, 16]. The infinite-valued variant— called simply “Łukasiewicz logic”, or  $\mathbf{L}$ — is a member of the family of t-norm based fuzzy logics [14, 10].

---

<sup>1</sup> This formula is of philosophical interest. See Remark 6 in [20].

**Definition 1** ( $\mathbf{L}$ ). *The Hilbert system for  $\mathbf{L}$  has four schematic axioms,*

$$\begin{aligned} & A \supset (B \supset A) \\ & (A \supset B) \supset ((B \supset C) \supset (A \supset C)) \\ & ((A \supset B) \supset B) \supset ((B \supset A) \supset A) \\ & ((B \supset \perp) \supset (A \supset \perp)) \supset (A \supset B) \end{aligned}$$

along with modus ponens. Other connectives are defined as

$$\begin{aligned} \neg A &=_{def} A \supset \perp & \top &=_{def} \neg \perp \\ A \vee B &=_{def} (A \supset B) \supset B & A \oplus B &=_{def} \neg A \supset B \\ A \wedge B &=_{def} \neg(\neg A \vee \neg B) & A \odot B &=_{def} \neg(\neg A \oplus \neg B) \end{aligned}$$

with the notation for iterated connectives defined for  $n \in \mathbb{N}$  as

$$\begin{aligned} 0 \cdot A &=_{def} \perp & A^0 &=_{def} \top \\ (n+1) \cdot A &=_{def} A \oplus (n \cdot A) & A^{n+1} &=_{def} A \odot A^n \end{aligned}$$

A detailed discussion of  $\mathbf{L}$  can be found in [12].

### 3 MV-Algebras

MV-algebras were introduced in [4, 5] as an algebraic counterpart to show the completeness on Łukasiewicz logics.

**Definition 2.** *An MV- (for “many-valued”) algebra is a structure  $\langle M, \oplus, \neg, 0 \rangle$  where  $\langle M, \oplus, 0 \rangle$  is a commutative monoid, with operators defined from  $0, \oplus$  and  $\neg$  as*

$$\begin{aligned} 1 &=_{def} \neg 0 \\ x \supset y &=_{def} \neg x \oplus y & x \odot y &=_{def} \neg(\neg x \oplus \neg y) \\ x \vee y &=_{def} (x \supset y) \supset y & x \wedge y &=_{def} \neg(\neg x \vee \neg y) \end{aligned}$$

and satisfying the conditions

$$\begin{aligned} \neg \neg x &= x \\ x \oplus 1 &= 1 \\ x \vee y &= y \vee x \end{aligned}$$

We define iterated operators for  $n \in \mathbb{N}$  as

$$\begin{aligned} 0 \cdot x &=_{def} 0 & x^0 &=_{def} 1 \\ (n+1) \cdot x &=_{def} (n \cdot x) \oplus x & x^{n+1} &=_{def} x^n \odot x \end{aligned}$$

**Definition 3 (Ordering Relation).** *The ordering relation for an MV-algebra  $M$  is defined for  $x, y \in M$  as  $x \leq y$  if and only if  $x \vee y = y$ . We call an MV-algebra **linearly ordered** (or “linear” for short) if and only if for all  $x, y \in M$ , either  $x \leq y$  or  $y \leq x$ .*

*Remark 1.* It can be shown that an MV-algebra is a lattice bounded by 0 and 1, with  $\vee$  and  $\wedge$  as the join and meet operators, respectively.

*Example 1.* The set of reals in  $[0, 1]$  with the mappings  $\mu(x \oplus y) = \min\{1, \mu(x) + \mu(y)\}$ ,  $\mu(\neg x) = 1 - \mu(x)$  and  $\mu(0) = 0$ . is an MV-Algebra.

**Lemma 1 (Chang Representation Theorem).** *Every MV-algebra is isomorphic to a subdirect product of linear MV-algebras.*

*Proof.* See [5].

*Remark 2.* It is a result in the theory of *universal algebra* that the subdirect product of a class of algebras shares the same equations as that class of algebras [3]. So if an equation holds for all linear MV-algebras, then it holds for all MV-algebras.

*Remark 3.* The translation between formulae in the language of  $\mathbf{L}$  and terms from the theory of MV-algebras is straightforward:

$$\begin{array}{ll} \tau(x) = x \text{ for atomic } x & \tau^{-1}(P) = P \text{ for propositional variable } P \\ \tau(\neg x) = \tau(x) \supset \perp & \tau^{-1}(A \supset \perp) = \neg \tau^{-1}(A) \\ \tau(x \supset y) = \tau(x) \supset \tau(y) & \tau^{-1}(A \supset B) = \tau^{-1}(A) \supset \tau^{-1}(B) \\ \tau(0) = \perp & \tau^{-1}(\perp) = 0 \end{array}$$

**Lemma 2 (Completeness).** *A formula  $A$  is valid in  $\mathbf{L}$  iff  $\tau^{-1}(A) = 1$  in every MV-algebra, where  $\tau^{-1}(A)$  is a translation of formula  $A$  into a term from the theory of MV-algebras given in Remark 3.*

*Proof.* See [5].

## 4 Two Equations in MV-Algebras

Below we outline the proof of two equations in MV-algebras and their corresponding formulae in  $\mathbf{L}$ . A more detailed proof can be found in [20].

**Lemma 3.** *Let  $x, y$  be elements of an MV-algebra  $M$ , and  $n \in \mathbb{N}$ . If  $x \leq y$ , then  $x^n \leq y^n$ .*

*Proof.* By induction on  $n$ .

*The base case ( $n = 0$ ) is trivial. For the induction step, assume  $x^{n-1} \leq y^{n-1}$ . Then  $x^{n-1} \odot x \leq y^{n-1} \odot x$  and  $y^{n-1} \odot x \leq y^{n-1} \odot y$  (by monotonicity). So  $x^{n-1} \odot x \leq y^{n-1} \odot y$ , that is,  $x^n \leq y^n$  (by transitivity).  $\square$*

**Theorem 1.** *Let  $x, y$  be elements of an MV-algebra  $M$ , and  $n \in \mathbb{N}$ . Then  $x^n \vee y^n = (x \vee y)^n$ .*

*Proof.* We first assume that the MV-algebra is linear.

If  $x \leq y$ , then  $x^n \vee y^n = y^n$  (Lemma 3). And  $(x \vee y)^n = (y)^n = y^n$  (by substitution). So when  $x \leq y$ , then  $x^n \vee y^n = (x \vee y)^n$ .

Similarly when  $y \leq x$ .

So for any linear MV-algebra,  $x^n \vee y^n = (x \vee y)^n$ .

From the Chang Representation Theorem (Lemma 1) and universal algebra (Remark 2), this equation holds for all MV-algebras.  $\square$

**Corollary 1.** Let  $x, y$  be elements of an MV-algebra  $M$ , and  $n \in \mathbb{N}$ .

Then  $(n \cdot x) \wedge (n \cdot y) = n \cdot (x \wedge y)$ .

*Proof.* Dual of Theorem 1.  $\square$

**Theorem 2.** The following are valid formulae in  $\mathcal{L}$ , for  $n \in \mathbb{N}$

$$A^n \vee B^n \supset (A \vee B)^n \quad (1)$$

$$(A \vee B)^n \supset A^n \vee B^n \quad (2)$$

$$n \cdot (A \wedge B) \supset (n \cdot A) \wedge (n \cdot B) \quad (3)$$

$$(n \cdot A) \wedge (n \cdot B) \supset n \cdot (A \wedge B) \quad (4)$$

*Proof.* Straightforward, from completeness (Lemma 2).  $\square$

We will refer to the formulae (1) and (3) as the *easy* formulae, and their converse formulae (2) and (4) as the *hard* formulae.

## 5 Comments on Proofs of Easy and Hard Formulae

Before we discuss proofs of these formulae, we first note the following points:

- For each  $n$ , the easy and hard formulae are the converse of one another, hence they have the same size and overall complexity.
- The size of the formulae grow linearly with  $n$ . Likewise, when the formulae are expressed using only the connectives  $\supset$  and  $\perp$ , the size grows linearly with  $n$  (they are larger than those with other connectives by a constant multiple).
- However, the formulae expressed in terms of  $\supset$  and  $\perp$  are very complex, even for low values of  $n$ . For example, formula (2) for  $n = 2$  is

$$\begin{aligned} & (((A \supset B) \supset B) \supset (((A \supset B) \supset B) \supset \perp)) \supset \perp) \supset \\ & (((A \supset (A \supset \perp)) \supset \perp) \supset ((B \supset (B \supset \perp)) \supset \perp)) \supset ((B \supset (B \supset \perp)) \supset \perp)) \end{aligned}$$

The implications in such formulae are deeply nested, which makes them useful for evaluating how well proof systems handle such complicated formulae.

We are unaware of any Hilbert-style proofs of these formulae, so we cannot discuss how such proofs would be structured. But we can illustrate some proof-theoretic ideas about the distinction between easy and hard formulae by looking at the proofs in various Gentzen-style calculi.

**Definition 4 (LSC).**

$$\begin{array}{c}
\overline{A \Rightarrow A} \quad \overline{\Rightarrow \top} \quad \overline{\perp \Rightarrow} \\
\frac{\Gamma \Rightarrow \Delta}{\Gamma, A \Rightarrow \Delta} LW \quad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow A, \Delta} RW \\
\frac{\Gamma \Rightarrow A, \Delta \quad \Gamma', B \Rightarrow \Delta'}{\Gamma, \Gamma', A \supset B \Rightarrow \Delta, \Delta'} L\supset \quad \frac{\Gamma, A \Rightarrow B, \Delta}{\Gamma \Rightarrow A \supset B, \Delta} R\supset \\
\frac{\Gamma, A, B \Rightarrow \Delta}{\Gamma, A \odot B \Rightarrow \Delta} L\odot \quad \frac{\Gamma \Rightarrow A, \Delta \quad \Gamma' \Rightarrow B, \Delta'}{\Gamma, \Gamma' \Rightarrow A \odot B, \Delta, \Delta'} R\odot \\
\frac{\Gamma, A \Rightarrow \Delta \quad \Gamma, B \Rightarrow \Delta}{\Gamma, A \vee B \Rightarrow \Delta} LV \quad \frac{\Gamma \Rightarrow A, \Delta}{\Gamma \Rightarrow A \vee B, \Delta} RV_1 \quad \frac{\Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow A \vee B, \Delta} RV_2 \\
\frac{\Gamma, (A \supset B) \supset B \Rightarrow A, B, \Delta}{\Gamma, (A \supset B) \supset B \Rightarrow A \vee B, \Delta} RV_c \quad \frac{\Gamma \Rightarrow \Delta, A \quad A, \Gamma' \Rightarrow \Delta'}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'} Cut
\end{array}$$

where  $\Gamma, \Gamma', \Delta, \Delta'$  are multisets of formulae. The rules for  $\oplus$  and  $\wedge$  are the duals of the rules for  $\odot$  and  $\vee$ , respectively.

*Remark 4.* The  $L\supset$  and  $R\supset$  rules are derived using the definition  $A \supset B =_{def} \neg A \oplus B$ .

*Remark 5.* **LSC** is affine linear logic with the addition of the rules  $RV_c$  and  $L\wedge_c$  [7].

We start with the sequent calculus **LSC** from [7]. Proofs of the easy formulae (1) and (3) are straightforward in **LSC**. For formula (1), we have the following outline of a proof for  $n \geq 2$ :

$$\frac{\frac{\frac{A \Rightarrow A}{A \Rightarrow A \vee B} RV \quad \frac{\vdots}{A^{n-1} \Rightarrow (A \vee B)^{n-1}} R\odot}{A, A^{n-1} \Rightarrow (A \vee B)^n} L\odot \quad \frac{\vdots}{B^n \Rightarrow (A \vee B)^n} LV}{A^n \vee B^n \Rightarrow (A \vee B)^n} LV$$

There is a similar proof for (3). Clearly, the depth of the proof is bounded by  $n$ .

In contrast, we cannot derive the hard formulae without the use of cut. For  $n = 2$ , we note the derivation fragment

$$\frac{\frac{\frac{A \Rightarrow A \quad A \Rightarrow A}{A, A \Rightarrow A^2} R\odot \quad \frac{\vdots}{A, B \Rightarrow A^2 \vee B^2} ?}{A, A \Rightarrow A^2 \vee B^2} RV \quad \frac{\vdots}{B, A \vee B \Rightarrow A^2 \vee B^2} LV}{A, A \vee B \Rightarrow A^2 \vee B^2} LV \quad \frac{A \vee B, A \vee B \Rightarrow A^2 \vee B^2}{(A \vee B)^2 \Rightarrow A^2 \vee B^2} L\odot} LV$$

where  $A, B \Rightarrow A^2 \vee B^2$  cannot be derived without cut—this is not surprising as **LSC** is known not to be cut-free [7]. We can see that any derivation of  $(A \vee B)^n \Rightarrow A^n \vee B^n$

depends on derivations of  $A^j, B^k \Rightarrow A^n \vee B^n$  for all  $1 \leq j, k \leq (n-1)$  such that  $j+k = n$ , some of which require cut.<sup>2</sup> We have yet to find a proof for these sequents in **LSC**.

Alternatively, we can examine the cut-free hypersequent calculus **GL** from [17]. Proofs of the easy formulae for  $n = 2$  can be found which do not contain instances of the multi-component *EC* or *S* rules, so could easily be converted to the sequent calculus variant **GL<sub>s</sub>** from [17]. (We used experimental implementations of the calculus, and a variant derived for more efficient goal-directed proof search [19]. The proofs are too large to include in this paper.)

**Definition 5 (GL).**

$$\begin{array}{c} \Rightarrow \quad \overline{A \Rightarrow A} \quad \overline{\perp \Rightarrow A} \\ \frac{\mathcal{H} \mid \Gamma, B \Rightarrow A, \Delta}{\mathcal{H} \mid \Gamma, A \supset B \Rightarrow \Delta} L\supset \quad \frac{\mathcal{H} \mid \Gamma, A \Rightarrow B, \Delta \quad \mathcal{H} \mid \Gamma \Rightarrow \Delta}{\mathcal{H} \mid \Gamma \Rightarrow A \supset B, \Delta} R\supset \\ \frac{\mathcal{H} \mid \Gamma \Rightarrow \Delta}{\mathcal{H} \mid \Gamma, A \Rightarrow \Delta} LW \quad \frac{\mathcal{H}}{\mathcal{H} \mid \Gamma \Rightarrow \Delta} EW \quad \frac{\mathcal{H} \mid \Gamma \Rightarrow \Delta \mid \Gamma \Rightarrow \Delta}{\mathcal{H} \mid \Gamma \Rightarrow \Delta} EC \\ \frac{\mathcal{H} \mid \Gamma_1 \Rightarrow \Delta_1 \quad \mathcal{H} \mid \Gamma_2 \Rightarrow \Delta_2}{\mathcal{H} \mid \Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2} M \quad \frac{\mathcal{H} \mid \Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2}{\mathcal{H} \mid \Gamma_1 \Rightarrow \Delta_1 \mid \Gamma_2 \Rightarrow \Delta_2} S \end{array}$$

where  $\Gamma, \Delta$  are multisets of formulae and  $\mathcal{H}$  refers to a hypersequent context of 0 or more sequents, called “components”. (An introductory survey of hypersequent calculi can be found in [1].)

*Remark 6.* [8] gives a variant of **GL** with an invertible variant of the  $L\supset$  rule:

$$\frac{\mathcal{H} \mid \Gamma, B \Rightarrow A, \Delta \mid \Gamma \Rightarrow \Delta}{\mathcal{H} \mid \Gamma, A \supset B \Rightarrow \Delta} L\supset'$$

We note that  $L\supset$  and  $L\supset'$  are interderivable.

Proofs of the hard formulae were difficult to find in **GL**. Because the implicational variants of the formulae are complex, we derived rules for the  $\vee$  and  $\odot$  operators:

$$\begin{array}{c} \frac{\mathcal{H} \mid \Gamma, A \Rightarrow \Delta \quad \mathcal{H} \mid \Gamma, B \Rightarrow \Delta}{\mathcal{H} \mid \Gamma, A \vee B \Rightarrow \Delta} L\vee \quad \frac{\mathcal{H} \mid \Gamma \Rightarrow \Delta \quad \mathcal{H} \mid \Gamma \Rightarrow B, \Delta \mid \Gamma \Rightarrow A, \Delta}{\mathcal{H} \mid \Gamma \Rightarrow A \vee B, \Delta} R\vee \\ \frac{\mathcal{H} \mid \Gamma, \perp \Rightarrow \Delta \quad \mathcal{H} \mid \Gamma, A, B \Rightarrow \Delta}{\mathcal{H} \mid \Gamma, A \odot B \Rightarrow \Delta} L\odot \quad \frac{\mathcal{H} \mid \Gamma \Rightarrow \Delta \quad \mathcal{H} \mid \Gamma \Rightarrow \perp, \Delta \mid \Gamma \Rightarrow A, B, \Delta}{\mathcal{H} \mid \Gamma \Rightarrow A \odot B, \Delta} R\odot \end{array}$$

Using these rules, a proof of a hard formula is fairly straightforward. However, these rules are of no help in proving the implicational forms of that formula.

<sup>2</sup> We have abused our notation slightly.  $A^j, B^k$  in the antecedent is equivalent to  $j$  instances of  $A$  and  $k$  instances of  $B$  in the antecedent. We note without proof that  $(A^j \odot B^k) \supset (A^{j+k} \vee B^{j+k})$  is a theorem of **L**.

We note that the proof of  $(A \vee B)^2 \Rightarrow A^2 \vee B^2$  with these derived rules contains a proof of  $A, B \Rightarrow A^2 \vee B^2$ , and that proof requires the use of the hypersequent  $S$  (split) rule:

$$\frac{\Rightarrow}{A, B \Rightarrow} LW^2 \frac{\begin{array}{c} \vdots \\ A, A, B, B \Rightarrow (A \odot A), (B \odot B) \end{array}}{A, B \Rightarrow (A \odot A) \mid A, B \Rightarrow (B \odot B)} \frac{S}{RV} \frac{}{A, B \Rightarrow ((A \odot A) \vee (B \odot B))}$$

The split rule  $S$  is not a desirable rule for goal-directed proof search when used with rules that have multiple components in their premisses, as the effect is akin to contraction. For example,

$$\frac{\begin{array}{c} \vdots \\ \mathcal{H} \mid \Gamma \Rightarrow \Delta \end{array} \quad \frac{\begin{array}{c} \vdots \\ \mathcal{H} \mid \Gamma, \Gamma \Rightarrow A, B, \Delta, \Delta \end{array}}{\mathcal{H} \mid \Gamma \Rightarrow B, \Delta \mid \Gamma \Rightarrow A, \Delta} S}{\mathcal{H} \mid \Gamma \Rightarrow A \vee B, \Delta} RV$$

We note that there is a terminating variant of  $\mathbf{GL}$  called  $\mathbf{GL}_t$  [17] that has a restricted form of the split rule which operates only on atomic formulae, but which multiplies the formulae in the premiss.

## 6 Benchmarks of Implementations

We now illustrate our relative labels of “easy” and “hard” with experimental data from implementations of theorem provers based on different systems in Table 1.

The theorem provers were implemented in SWI Prolog (version 5.6.33), in part because it has the `time` predicate which counts the number of inferences made when attempting to unify a goal—this number is constant for the same query across different machines. Tests were run on formulae (1) and (2) for  $n \geq 2$  with the optimisation flag enabled and all stack sizes set to the maximum values.

Where needed, the implementations used `clpr` (Constraint Logic Programming over Reals) library, which is a port of the CLPR system from Sicstus Prolog [9]. (Tests were also run using the rationals version of the library, `clpq`; aside from an issue noted below, the relative inference counts, though slightly higher, had similar ratios between easy and hard formulae, and growth rates relative to  $n$ , as the `clpr` library.)

The implementations we tested performed well on the axioms of  $\mathbf{L}$  and various well-known theorems from [2, 12]. Aside from the program given in [13], we are unaware of generally-available special-purpose theorem provers for  $\mathbf{L}$  to use in this report. We wrote our own implementations of theorem provers as a means of studying proofs of these formulae. The implementations are not optimised, and the numbers should not necessarily be seen as means of comparing the different systems.

For space considerations, we will omit explicit descriptions of the methods used by the implemented proof systems. The reader is advised to refer to cited papers. However, source code for the implementations discussed in this report are available from the author’s website, or by request.

**Table 1.** Comparative Prolog inference counts for tableaux implementations.

$n$	Inference Count					
	Hähnle		Olivetti		Ciabattoni et al.	
	(1)	(2)	(1)	(2)	(1)	(2)
2	1,957,210	1,804,133	44,403	101,236	16,573	20,403
3	4,891,223	6,398,578	236,499	1,100,768	91,104	81,497
4	13,317,246	20,886,776	798,593	6,818,706	558,343	262,262
5	38,114,941	70,383,639	2,244,775	37,283,340	3,648,396	770,151
6	118,661,664	254,427,842	5,600,001	170,602,568	*	2,123,454
7	401,007,825	984,466,176	13,437,857	710,355,342	*	5,612,699
8	1,444,053,450	4,002,546,748	31,073,217	2,670,896,378	*	14,389,746

The Hähnle column refers to a port of the Eclipse Prolog program given in [13] which uses `c1pr` library. The proof method is a Mixed Integer Program (MIP) representation of the query formula to check the satisfiability for values less than 1. All formulae are analysed in terms of the  $\oplus$  and  $\neg$  connectives (which is roughly the same as being given in terms of  $\supset$  and  $\perp$ ), which may be one reason why the inference counts are significantly larger than the implementations of Olivetti and Ciabattoni et al. proof systems. The hard formulae require more inferences to validate than the easy formulae (for  $n \geq 3$ ) in the Hähnle implementation. However, as  $n$  increases, the ratio of inferences for validating the hard formulae against the easy formulae appears to increase *linearly*.

The Olivetti column refers to an implementation of the tableaux method given in [18] (using the improved rules from §5). The system is based on the Kripke semantics for  $\mathbf{L}$  given in [21, 22], and branches are closed using linear programming (LP) methods. Our implementation also uses the `c1pr` library to check the constraints of labels as each operator is analysed. There are primitive (non-derived) rules for the  $\vee$  and  $\wedge$  operators in the system, but not for the  $\oplus$  and  $\odot$  operators, which had to be converted to  $\supset$  and  $\perp$  representations by the implementation. This implementation does not generate a proof tree. We note that the hard formulae also consistently require more inferences to validate than the easy formulae. Furthermore, as  $n$  increases, the ratio appears to increase *exponentially*.

The Ciabattoni et al. column refers to an implementation of a hypersequent of relations<sup>3</sup> calculus given in [6] (specifically the Co-NP calculus from §5). The system analyses formulae into disjunctive multisets of inequalities, and has rules which are similar to the rules for  $\mathbf{GL}$  with the derived rules for  $\vee$  and  $\odot$  given above. When all the formulae are analysed into atomic terms, the lists are transformed into conjunctions of negated inequalities, and LP methods are used to check the unsatisfiability of those lists. This system has primitive rules for the  $\vee$  and  $\odot$  operators, so no conversion of formulae was needed for the tests. The `c1pr` library was also used in this implementation. The \* in the table indicate where the tests failed— we suspect this may be due

<sup>3</sup> A hypersequent of relations is a hypersequent where the  $\Rightarrow$  operator in components can be either  $\geq$  or  $<$ .

to problems with the constraint library, since when the rationals library was used in place of the reals library, the case for  $n = 6$  succeeded, although the cases for  $n \geq 7$  failed in that version. We note that unlike the other implementations, the easy theorems required more inferences than the hard theorems (but like the other implementations, the ratio of number of inferences also appears to increase at an exponential rate with  $n$ ). We suspect this is because the *duals* of the inequalities are checked— however, for the tests on implicational variants of the formulae (below), the hard formulae do require more inferences. We are unsure why the ratio is reversed when implicational formulae are used.

We ran a second set of the tests on the implementations of the Olivetti and Ciabattoni et al. proof systems using implicational variants of the formulae. The results are in Table 2. The numbers for the Hähnle implementation are included for comparison, since the  $\oplus$  and  $\neg$  forms of formulae are similar to the implicational forms. We note that in all cases, the hard formulae generally require more inferences to validate than the easy formulae. (The \*\* in the table indicates a stack overflow occurred.) What is interesting is that as  $n$  increases, the ratio of inferences for hard formulae to easy formulae appears to increase at a much smaller rate than tests for non-implicational formula.

**Table 2.** Comparative Prolog inference counts for implicational versions of formulae

$n$	Inference Count					
	Hähnle		Olivetti		Ciabattoni et al.	
	(1)	(2)	(1)	(2)	(1)	(2)
2	1,957,210	1,804,133	671,839	805,213	2,002,182	2,542,614
3	4,891,223	6,398,578	10,526,995	19,173,190	46,268,462	74,514,652
4	13,317,246	20,886,776	107,144,035	206,458,981	785,191,785	1,396,900,149
5	38,114,941	70,383,639	966,442,327	1,719,263,455	11,726,123,678	20,676,641,593
6	118,661,664	254,427,842	12,323,671,151	8,570,932,566	**	**

Our implementations of **GL** and related symbolic calculi generally required too much time even to prove the easy formulae in their implication forms for  $n \geq 3$ . For example, the terminating calculus **GL<sub>t</sub>**, prove (1) for  $n = 2$  in 28,083 inferences, but for  $n = 3$  in 293,361,238 inferences. We believe this is because of the complexity added by multiple components in the premisses of hypersequent rules. (We note that the implementation of theorem provers based on syntactic systems for Łukasiewicz logic is not considered practical in lieu of the existence of more efficient constraint-based solvers, e.g. for hypersequent calculi [11].)

However, our implementation of the terminating calculus **GL<sub>t</sub>** did not find a proof for (2) for  $n = 2$ . We also wrote and tested implementations of **GL<sub>l</sub>**, a labelled variant of the hypersequent calculus **GL** [17], which eliminates the structural rules and validates axioms by solving for constraints on the labels. These implementations also did not find proofs of the formula.<sup>4</sup> (Indeed, this shared inability to prove these theorems inspired

<sup>4</sup> We note that these implementations were also unable to find proofs for  $(A \odot B) \supset (A^2 \vee B^2)$ .

this paper.) We are reviewing whether this is due to misinterpretations of the calculi or other implementation issues.

In conclusion, our experience suggests that these formulae are useful for testing and benchmarking automated theorem provers for  $\mathbf{L}$ .

## 7 Future Work

We would like to investigate the properties of proofs of the hard formulae, and whether there are techniques for reducing the complexity of proof search for these formulae. The reversal of easy versus hard problems for the Ciabattoni et al. calculus [6] when converting formulae into implication form needs further study. This may give clues on ways of converting the hard formulae into easier problems.

We are also interested in the relationship (if there is one) between the *Cut* rule in calculi where it is primitive, and the *S* rule in hypersequent calculi which are cut-free, and in how *S* affects the permutability of invertible rules in hypersequent systems.

We would like to incorporate ideas from the terminating calculus  $\mathbf{GL}_t$ , the syntactic variant of the Ciabattoni et al. calculus called  $\mathbf{rHL}$  (which applies structural hypersequent rules to atomic formulae instead of solving for constraints) and the mix-free calculus in [19] to derive a symbolic calculus which can be implemented with comparable in efficiency to the semantic theorem provers for  $\mathbf{L}$ .

As part of a project to investigate the use of hypersequents as a basis for automated proof search, we will investigate implementation of techniques to identify and eliminate redundant components during the search in an efficient manner.

## 8 Acknowledgements

We would like to thank Agata Ciabattoni and George Metcalfe for answering questions about  $\mathbf{GL}$ , Reiner Hähnle for providing a copy of his paper, and Roy Dyckhoff, James McKinna and anonymous referees for their feedback on earlier drafts of this report.

This work has been partially supported by The EmBounded Project (IST-510255), a three-year FET-Open collaborative project funded by the European Union Framework 6 Programme.

## References

1. A. Avron. The method of hypersequents in the proof theory of propositional non-classical logics. In W. Hodges, M. Hyland, C. Steinhorn, and J. Truss, editors, *Logic: Foundations to Applications*. Oxford Science Pub., 1996.
2. G. Beavers. Distribution in Lukasiewicz logics. *Bull. Sec. Log.*, 21(4):140–146, 1992.
3. S.N. Burris and H.P. Sankappanavar. *A Course in Universal Algebra*. Springer-Verlag, New York, Millenium edition, 1981, 1999.
4. C.C. Chang. Algebraic analysis of many valued logics. *Trans. Amer. Math. Soc.*, 88(2):467–490, July 1958.
5. C.C. Chang. A new proof of the completeness of the Lukasiewicz axioms. *Trans. Amer. Math. Soc.*, 93(1):74–80, October 1959.

6. A. Ciabattani, C.G. Fermüller, and G. Metcalfe. Uniform rules and dialogue games for fuzzy logics. In F. Baader and A. Voronkov, editors, *LPAR 2004*, volume 3452 of *Lecture Notes in Computer Science*, pages 496–510. Springer, 2004.
7. A. Ciabattani and D. Luchi. Two connections between linear logic and Łukasiewicz logics. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Kurt Gödel Colloquium*, volume 1289 of *Lecture Notes In Computer Science*, pages 128–139, Berlin / Heidelberg, 1997. Springer.
8. A. Ciabattani and G. Metcalfe. Bounded Łukasiewicz logics. In M. Cialdea Mayer and F. Pirri, editors, *TABLEAUX 2003*, volume 2796 of *Lecture Notes in Computer Science*, pages 32–47. Springer, 2003.
9. L. De Koninck and K.U. Leuven. *SWI Prolog Manual*, chapter A.14. 2007. Online. URL=<http://gollem.science.uva.nl/SWI-Prolog/Manual/clpqr.html> ).
10. F. Esteva and L. Godo. Monoidal t-norm based logic: towards a logic for left-continuous t-norms. *Fuzzy Sets and Systems*, 124(3):271–288, December 2001.
11. D. Gabbay, G. Metcalfe, and N. Olivetti. Hypersequents and fuzzy logic. *Revista de la Real Academia de Ciencias*, 98(1):113–126, 2004.
12. S. Gottwald. *A Treatise on Many-Valued Logics*. Research Studies Press, Baldock, Hertfordshire, England, 2001.
13. R. Hähnle. Proof theory of many-valued logic–linear optimization–logic design: connections and interactions. *Soft Comput.*, 1(3):107–119, 1997.
14. P. Hájek. *Metamathematics of Fuzzy Logics*. Kluwer Academic Pub., London, 1998.
15. J. Łukasiewicz. Philosophische Bemerkungen zu mehrwertigen Systemen des Aussagenkalküls (Philosophical remarks on many-valued systems of propositional logics). In S. McCull, editor, *Polish Logic 1920-1939*. Clarendon Press, Oxford, 1967.
16. J. Łukasiewicz and A. Tarski. Untersuchungen über den Aussagenkalkül (Investigations into the sentential calculus). In A. Tarski, J. Corcoran, and J.H. Woodger, editors, *Logic, Semantics, Metamathematics: Papers from 1923 to 1938*, pages 38–59. 1983.
17. G. Metcalfe, N. Olivetti, and D. Gabbay. Sequent and Hypersequent Calculi for Abelian and Łukasiewicz Logics. *ACM Transactions on Computational Logic*, 6(3):578–613, 2005.
18. N. Olivetti. Tableaux for Łukasiewicz Infinite-valued Logic. *Studia Logica*, 73(1):81–111, February 2003.
19. R. Rothenberg. An hypersequent calculus for Łukasiewicz logic without the merge rule. In *Proc. of the Thirteenth Workshop on Automated Reasoning (ARW 2006), University of Bristol, Bristol, England, 3-4 April 2006*, pages 33–34, 2006.
20. R. Rothenberg. Distribution theorems in MV-algebras. Technical report, School of Computer Science, University of St Andrews, January 2007. Online. URL=<http://www.cs.st-andrews.ac.uk/~rr/pubs/mvdist.pdf>).
21. D. Scott. Completeness and axiomatizability in many-valued logic. In L. Henkin et al., editor, *Proc. of Symposia in Pure Mathematics Volume XXV: Proceedings of the Tarski Symposium*, pages 411–435. AMS, 1974.
22. A. Urquhart. Many-valued logic. In Dov M. Gabbay and Franz Guenther, editors, *Handbook of Philosophical Logic*, volume 3, pages 71–116. D. Reidel Publishing Company, Lancaster, 1986.