Varieties of Generalized Hoops and Integral GBL-algebras

Peter Jipsen

Chapman University, Orange, California

AMS Section Meeting, April 13, 2013, Boulder Colorado
Generalized Hoops

Generalized hoops were first studied by Bosbach [1969, 70] and the name **hoop** was introduced by Büchi and Owen [1975].

A **generalized hoop** \((A, \cdot, 1, \backslash, /)\) is a residuated partially ordered monoid in which

\[
x \leq y \iff \exists u (x = uy) \iff \exists v (x = yv).
\]

i.e. the monoid is **naturally ordered**, hence **integral**: \(x \leq 1\)

**Residuated** means: \(xy \leq z \iff y \leq x \backslash z \iff x \leq z / y\)
Two simple identities

\[
\frac{\frac{x}{y}}{z} = \frac{x}{zy}
\]
Two simple identities

\[
\frac{(\frac{x}{y})}{z} = \frac{1}{z} (\frac{x}{y}) =
\]
Two simple identities

\[
\left(\frac{x}{y}\right) \frac{1}{z} = \frac{x}{zy}
\]
Two simple identities

\[
\frac{\left(\frac{x}{y}\right)}{z} = \frac{1}{z} \left(\frac{x}{y}\right) = \frac{x}{zy}
\]

\[
(x/y)/z =
\]
Two simple identities

$$\frac{\frac{x}{y}}{z} = \frac{1}{\frac{z}{y}} = \frac{x}{zy}$$

$$(x/y)/z = x/(zy)$$
Two simple identities

\[
\frac{\left(\frac{x}{y}\right)}{z} = \frac{1}{z}\left(\frac{x}{y}\right) = \frac{x}{zy}
\]

\[
\frac{x}{y}/z = x/(zy)
\]

\[
x \backslash (y \backslash z) = \]

Two simple identities

\[
\frac{\left(\frac{x}{y}\right)}{z} = \frac{1}{z} \left(\frac{x}{y}\right) = \frac{x}{zy}
\]

\[
\frac{x}{y}/z = x/(zy)
\]

\[
x \setminus (y \setminus z) = (y
\]
Two simple identities

$$\frac{\frac{x}{y}}{z} = \frac{1}{z} \left(\frac{x}{y}\right) = \frac{x}{zy}$$

$$\frac{x}{y}/z = x/(zy)$$

$$x \backslash (y \backslash z) = (yx)$$
Two simple identities

\[
\frac{\frac{x}{y}}{z} = \frac{1}{z} \left( \frac{x}{y} \right) = \frac{x}{zy}
\]

\[
\frac{x}{y} / z = x / (zy)
\]

\[
x \backslash (y \backslash z) = (yx) \backslash z
\]
Other simple identities

\[ \frac{x}{x} = 1 \quad \text{(true in integral residuated monoids)} \quad 1y = y \]

Therefore \[ \frac{x}{x} y = y \]

Another Basic identity: \( (x/y)y = (y/x)x \)

NOT true in residuated monoids, but an axiom of hoops.

Equivalent to \( x \leq y \Rightarrow x = (x/y)y \)

Equivalent to naturally ordered: \( x \leq y \Rightarrow \exists u(x = uy) \)
A lemma

If \( y = (x/x)y \) and \( x/(y \cdot z) = ((x/z)/y) \) and

\( (x/y)y = (y/x)x \) then \( \cdot \) is associative.

**Proof:** \( x(yz) = [((xy)z)/((xy)z)](x(yz)) \)
A lemma

If \( y = (x/x)y \) and \( x/(y \cdot z) = ((x/z)/y) \) and 
\( (x/y)y = (y/x)x \) then \( \cdot \) is associative.

**Proof:** \( x(yz) = (((xy)z)/((xy)z))(x(yz)) \)
\( = ((((xy)z)/z)/(xy))(x(yz)) \)
A lemma

If \( y = (x/x)y \) and \( x/(y \cdot z) = ((x/z)/y) \) and

\( (x/y)y = (y/x)x \) then \( \cdot \) is associative.

**Proof:** \( x(yz) = [(((xy)z)/(xy))z](x(yz)) \)

\[ = [(((xy)z)/z)/(xy)](x(yz)) \]

\[ = [(((xy)z)/z)/y]/x](x(yz)) \]
A lemma

If \( y = (x/x)y \) and \( x/(y \cdot z) = ((x/z)/y) \) and

\[(x/y)y = (y/x)x\] then \( \cdot \) is associative.

**Proof:**

\[x(yz) = \left[\left(\left(\left(\left(x(y)z\right)/\left((xy)z\right)\right)\right)/\left((x(y)z)/\left(xy)\right)\right)\right]\]

\[= \left[\left(\left(\left(\left(x(y)z\right)/\left(z\right)\right)/\left(xy\right)\right)\right)/\left(\left(x(y)z\right)/\left(xy\right)\right)\right]\]

\[= \left[\left(\left(\left(\left(x(y)z\right)/\left((y)z\right)\right)/\left(x\right)\right)\right)/\left(\left(x(y)z\right)/\left(xy\right)\right)\right]\]

\[= \left[\left(\left(\left(\left(x(y)z\right)/\left((y)z\right)\right)/\left(x\right)\right)\right)/\left(\left(x(y)z\right)/\left(xy\right)\right)\right]\]
A lemma

If \( y = (x/x)y \) and \( x/(y \cdot z) = ((x/z)/y) \) and
\( (x/y)y = (y/x)x \) then \( \cdot \) is associative.

**Proof:** \( x(yz) = [(((xy)z)/(xy)z)](x(yz)) \)
\[
= [(((xy)z)/z)/(xy)](x(yz))
\]
\[
= [(((xy)z)/z)/y]/x](x(yz))
\]
\[
= [(((xy)z)/(yz))/x](x(yz))
\]
\[
= [(((xy)z)/(x(yz)))](x(yz))
\]
A lemma

If \( y = (x/x)y \) and \( x/(y \cdot z) = ((x/z)/y) \) and

\( (x/y)y = (y/x)x \) then \( \cdot \) is associative.

**Proof:**

\[
x(yz) = [(((xy)z)/(xy))(x(yz))]
\]

\[
= [(((xy)z)/z)/(xy)](x(yz))
\]

\[
= [((((xy)z)/z)/y)/x](x(yz))
\]

\[
= [(((xy)z)/(yz))/x](x(yz))
\]

\[
= [(((xy)z)/(x(yz))][x(yz))
\]

\[
= [(x(yz))/(xy)(z)]((xy)z) =
\]
A lemma

If \( y = (x/x)y \) and \( x/(y \cdot z) = ((x/z)/y) \) and

\((x/y)y = (y/x)x\) then \( \cdot \) is associative.

**Proof:** \( x(yz) = [(((xy)z)/(xy)z)](x(yz)) \)

\[ = [(((xy)z)/z)/(xy)](x(yz)) \]

\[ = [(((xy)z)/z)/(y)/x](x(yz)) \]

\[ = [(((xy)z)/(yz))/x](x(yz)) \]

\[ = [(((xy)z)/(xyz))](x(yz)) \]

\[ = [(x(yz))/(x(yz))][(xy)z] = \text{ reverse steps to get } = (xy)z \]
Equational basis for generalized hoops

\[ x1 = x \]

\[ x/x = 1 = x\backslash x \]

\[ x/(yz) = (x/z)/y \quad y\backslash(z\backslash x) = (zy)\backslash x \]

\[ (x/y)y = (y/x)x = y(y\backslash x) \]

Generalized hoops are also called **pseudo hoops**

Note: The term \((x/y)y\) defines a binary operation that is **commutative** and **idempotent** \(((x/x)x = 1x = x)\).
Lemma: \((x/y)y\) is associative, hence written as \(x \land y\). It is a meet since \(x \leq y \iff 1 = y/x \iff x = (x/y)y\)

Proof. \((x \land y) \land z = (((x/y)y)/z)z\)
A meet-semilattice term

**Lemma**: $(x/y)y$ is associative, hence written as $x \land y$. It is a meet since $x \leq y \iff 1 = y/x \iff x = (x/y)y$

Proof. $(x \land y) \land z = (((x/y)y)/z)z = (z/(x/y)y)(x/y)y$
Lemma: \((x/y)y\) is associative, hence written as \(x \land y\). It is a meet since \(x \leq y \iff 1 = y/x \iff x = (x/y)y\)

Proof. \((x \land y) \land z = (((x/y)y)/z)z = (z/(x/y)y)(x/y)y\)

\[= ((z/y)/(x/y))(x/y)y\]
Lemma: \((x/y)y\) is associative, hence written as \(x \land y\). It is a meet since \(x \leq y \iff 1 = y/x \iff x = (x/y)y\)

Proof. \((x \land y) \land z = (((x/y)y)/z)z = (z/(x/y)y)(x/y)y\)

\[= ((z/y)/(x/y))(x/y)y\]

\[= ((x/y)/(z/y))(z/y)y\]
A meet-semilattice term

**Lemma:** \((x/y)y\) is associative, hence written as \(x \land y\). It is a meet since \(x \leq y \iff 1 = y/x \iff x = (x/y)y\)

Proof. \((x \land y) \land z = (((x/y)y)/z)z = (z/(x/y)y)(x/y)y\)

\[= ((z/y)/(x/y))(x/y)y\]

\[= ((x/y)/(z/y))(z/y)y\]

\[= (x/(z/y)y)(z/y)y\]
A meet-semilattice term

**Lemma:** \((x/y)y\) is associative, hence written as \(x \land y\). It is a meet since \(x \leq y \iff 1 = y/x \iff x = (x/y)y\)

Proof. \((x \land y) \land z = (((x/y)y)/z)z = (z/(x/y)y)(x/y)y\)

\[= ((z/y)/(x/y))(x/y)y\]

\[= ((x/y)/(z/y))(z/y)y\]

\[= (x/(z/y)y)(z/y)y = x \land (z \land y) = x \land (y \land z)\]
Multiplication distributes over meet

[Galatos ~04] Generalized hoops satisfy \((x \land y)z = xz \land yz\)

Preliminary: \(xz \leq xz \implies x \leq xz/z\)
Multiplication distributes over meet

[Galatos ~04] Generalized hoops satisfy \((x \land y)z = xz \land yz\)

Preliminary: \(xz \leq xz \implies x \leq xz/z\) hence \(xz \leq (xz/z)z\)
Multiplication distributes over meet

[Galatos ~04] Generalized hoops satisfy $(x \land y)z = xz \land yz$

Preliminary: $xz \leq xz \implies x \leq xz/z$ hence $xz \leq (xz/z)z$

$xz/z \leq xz/z \implies (xz/z)z \leq xz$
Multiplication distributes over meet

[Galatos ~04] Generalized hoops satisfy $(x \land y)z = xz \land yz$

Preliminary: $xz \leq xz \implies x \leq xz/z$ hence $xz \leq (xz/z)z$

$xz/z \leq xz/z \implies (xz/z)z \leq xz$ therefore $xz = (xz/z)z$
Multiplication distributes over meet

[Galatos ~04] Generalized hoops satisfy $(x \land y)z = xz \land yz$

Preliminary: $xz \leq xz \implies x \leq xz/z$  hence $xz \leq (xz/z)z$

$xz/z \leq xz/z \implies (xz/z)z \leq xz$  therefore $xz = (xz/z)z$

Now $(x \land y)z \leq xz \land yz$ always holds since $\cdot$ is order-preserving
Multiplication distributes over meet

[Galatos ~04] Generalized hoops satisfy \((x \land y)z = xz \land yz\)

Preliminary: \(xz \leq xz \implies x \leq xz/z\) hence \(xz \leq (xz/z)z\)

\(xz/z \leq xz/z \implies (xz/z)z \leq xz\) therefore \(xz = (xz/z)z\)

Now \((x \land y)z \leq xz \land yz\) always holds since \(\cdot\) is order-preserving

\(xz \land yz = (xz/yz)yz = ((xz/z)/y)yz\)
Multiplication distributes over meet

[Galatos ~04] Generalized hoops satisfy \((x \land y)z = xz \land yz\)

Preliminary: \(xz \leq xz \implies x \leq xz/z\) hence \(xz \leq (xz/z)z\)

\(xz/z \leq xz/z \implies (xz/z)z \leq xz\) therefore \(xz = (xz/z)z\)

Now \((x \land y)z \leq xz \land yz\) always holds since \(\cdot\) is order-preserving

\(xz \land yz = (xz/yz)yz = (((xz/z)/y)yz\)

\(= (y/((xz)/z))(xz/z)z\)
Multiplication distributes over meet

[Galatos ~04] Generalized hoops satisfy \((x \land y)z = xz \land yz\)

Preliminary: \(xz \leq xz \implies x \leq xz/z\) hence \(xz \leq (xz/z)z\)

\(xz/z \leq xz/z \implies (xz/z)z \leq xz\) therefore \(xz = (xz/z)z\)

Now \((x \land y)z \leq xz \land yz\) always holds since \(\cdot\) is order-preserving

\(xz \land yz = (xz/yz)yz = ((xz/z)/y)yz\)

\(= (y/((xz)/z))(xz/z)z = (y/((xz)/z))xz\)
Multiplication distributes over meet

[Galatos ~04] Generalized hoops satisfy \((x \land y)z = xz \land yz\)

Preliminary: \(xz \leq xz \implies x \leq xz/z\) hence \(xz \leq (xz/z)z\)

\(xz/z \leq xz/z \implies (xz/z)z \leq xz\) therefore \(xz = (xz/z)z\)

Now \((x \land y)z \leq xz \land yz\) always holds since \(\cdot\) is order-preserving

\(xz \land yz = (xz/yz)yz = ((xz/z)/y)yz\)

\[= (y/((xz)/z))(xz/z)z = (y/((xz)/z))xz\]

\[\leq (y/x)xz = (y \land x)z\]
Commutative generalized hoops are called **hoops**

In this case $x/y = y\backslash x$ usually written as $y \rightarrow x$

If we expand the signature of generalized hoops with $\lor$ and add lattice identities then we get **integral GBL-algebras**

Add **bottom 0**, **commutativity**, and $(x \rightarrow y) \lor (y \rightarrow x) = 1$

get Hajek’s **Basic Logic algebras**

Includes BA, Heyting algebras, MV-algebras, GA, PA

Open Problem: Is the equational theory of integral GBL-algebras decidable?
Finite generalized hoops

Finite GH are **reducts** of integral GBL-algebras

[J. & Montagna 06] Finite GBL-algebras are commutative

Hence finite GH are commutative

[J. & Montagna 09] Finite GBL-algebras are **poset products**

of **Wajsberg chains** $W_n = (\{0, a^{n-1}, \ldots, a^3, a^2, a, 1\}, \cdot, 1, \rightarrow)$

A poset product is a subalgebra of a direct product over a partially ordered index set
Poset products

For bounded GH or GBL-algebras $C_i$ indexed by a poset $P$

\[
\prod_{P} C_i = \{ f \in \prod_{i \in P} C_i : \forall i > j \in P \ (f(i) \neq 0 \implies f(j) = 1) \}
\]

The operations $\land, \lor, \cdot$ are defined pointwise and the bounds are the constant functions 0, 1. The residuals are given by

\[
(f \backslash g)(i) = \begin{cases} 
  f(i) \backslash g(i) & \text{if } f(j) \leq g(j) \text{ for all } j < i \\
  0 & \text{otherwise}
\end{cases}
\]

\[
(g / f)(i) = \begin{cases} 
  g(i) / f(i) & \text{if } f(j) \leq g(j) \text{ for all } j < i \\
  0 & \text{otherwise}
\end{cases}
\]
If the poset is **linear** we get an **ordinal sum** of the factors

If the poset is an **antichain**, we get the **direct product**

If the factors are **Boolean algebras**, get a **Heyting algebra**

Can build **all** finite GH and GBL-algebras: **pick a finite poset** $P$

**Pick a positive integer** $n_i$ for each $i \in P$

Get **all** finite GH and GBL-algebras uniquely up to isomorphism

The algebra is **subdirectly irreducible** iff poset has a **top**

Generalized hoops are **congruence distributive** [Botur, Dvurečenskij, Kowalski 2012]

Can construct lattice of **finitely generated subvarieties**
$W_m$ is a subalgebra of $W_n$ iff $m|n$

Therefore the varieties $V(W_n)$, ordered by inclusion, form the divisibility lattice $\mathbb{D}$

The lattice of all finitely generated subvarieties of Wasjberg hoops is isomorphic to the downset lattice of $\mathbb{D}$ [Komori 81]

**Theorem.** The poset of **finitely generated join irreducible BL-varieties** is isomorphic to $\mathbb{D}^* = \bigcup_{n=0}^{\infty} \mathbb{D}^n$ with the order on $\mathbb{D}^*$ extending the pointwise divisibility order on each component as follows: The order relation $(a_1, \ldots, a_m) \leq (b_1, \ldots, b_n)$ is a **covering relation** if and only if either

- $m = n$ and $(b_1, \ldots, b_n) = (a_1, \ldots, a_{i-1}, pa_i, a_{i+1}, \ldots, a_n)$ for some prime $p$ and a unique $i \leq n$, or
- $m + 1 = n$ and $(b_1, \ldots, b_n) = (a_1, \ldots, a_{i-1}, 1, a_i, \ldots, a_m)$ for some $i \in \{2, \ldots, n\}$
$D = \text{the divisibility lattice on } \mathbb{Z}^+$
SAT-solvers

SAT stands for *satisfiability* of Boolean formulas

Given a Boolean formula \( \varphi \) with propositional variables \( p_1, \ldots, p_n \)

decide if there is an assignment \( h : \{ p_1, \ldots, p_n \} \rightarrow \{ T, F \} \)
such that

\( h \) extended homomorphically to all formulas makes \( h(\varphi) = T \)

SAT was the first problem proved to be NP-complete

i.e., there is a nondeterministic Turing machine that decides
SAT in polynomial time and every other problem that can be
decided in nondeterministic polynomial time has a polynomial
time reduction to a SAT problem
SMT-solvers

SMT stands for *satisfiability modulo theories*

Combines SAT-solving with other decision procedures for fragments of first-order logic and arithmetic

**SMT-solvers** were developed in computer science for static analysis of programs

Input is a (limited) choice of a decidable theory and a list of Boolean combinations of atomic formulas in the signature of this theory
Quantifier-free decidable theories

**QF_LRA** quantifier free linear real number arithmetic with $+, -, <, =$

e.g. not($0 > x + y$ or $x + y > 5$) and $(x + x - y - y = 1)$

**QF_RA** is like QF_LRA but also allows multiplication, division

SMT-solvers decide if there exists an assignment of real numbers to the variables in the list of formulas such that all the formulas are true in $\mathbb{R}$; return assignment if it exists
How SMT-solvers work

Basic idea: replace atomic formulas by Boolean variables, call a SAT-solver

if the Boolean formulas are not satisfiable, return \( F \)

else use each possible Boolean assignment to generate a list of linear atomic formulas and call a Linear Programming package

if an assignment is found, return it, but if none of the Boolean assignments work, return \( F \)
SMT-solver input for abelian $\ell$-groups

Easy, the variety of abelian $\ell$-groups is generated by $(\mathbb{R}, \min, \max, +, -, 0)$

SMT_LIB2 is a standard LISP-like language for SMT-solver input

; Testing abelian l-group equations in SMT
(set-logic QF_LRA)
(define-fun wedge ((x Real) (y Real)) Real (ite (> x y) y x))
(define-fun vee ((x Real) (y Real)) Real (ite (> x y) x y))
(declare-const x Real)
(declare-const y Real)
(assert (> (vee (+ x x) (+ y y)) (+ (vee x y) (vee x y))))
; test if $(x + x) \lor (y + y) \leq (x \lor y) + (x \lor y)$ is an identity
(check-sat)
The idea of using SMT-solvers for logics based on intervals of the real numbers is from the following paper:


They give examples of SMT-LIB2 code for *Lukasiewicz logic* and *product logic*
SMT-solver input for MV-algebras

The variety of MV-algebras is $HSP(([0, 1], \wedge, \vee, \cdot, 1, 0, \to))$

; Testing MV-algebra equations in SMT
(set-logic QF_LRA)
(define-fun wedge ((x Real) (y Real)) Real (ite (> x y) y x))
(define-fun vee ((x Real) (y Real)) Real (ite (> x y) x y))
(define-fun oplus ((x Real) (y Real)) Real (wedge (+ x y) 1))
(define-fun cdot ((x Real) (y Real)) Real (vee (- (+ x y) 1) 0))
(define-fun neg ((x Real)) Real (- 1 x))
(define-fun to ((x Real) (y Real)) Real (wedge 1 (- (+ 1 y) x)))
(declare-const x Real) (assert (<= 0 x)) (assert (<= x 1))
(declare-const y Real) (assert (<= 0 y)) (assert (<= y 1))
(assert (< (to (vee (cdot x x) (cdot y y)) (cdot (vee x y) (vee x y)))) 1))
; test if $(x^2 \lor y^2) \to (x \lor y)^2 < 1$ is satisfiable
(check-sat)
Other standard Basic Logic algebras

For Gödel algebras redefine fusion as $\min(x,y)$.

\[(\text{define-fun cdot } ((x \text{ Real}) (y \text{ Real})) \text{ Real} (\text{ite } (> x y) y x))\]

For product algebras use

\[(\text{define-fun cdot } ((x \text{ Real}) (y \text{ Real})) \text{ Real} (\text{ite } (> x y) y x))\]
\[(\text{declare-const } x \text{ Real}) (\text{assert } (\leq x 0));\]
\[(\text{declare-const } x \text{ Real}) (\text{assert } (\leq x 0));\]

and do a translation to the formula that adds an extra variable $z$ (for bottom)

replacing variable $x$ by $x \lor z$ and subterms $s \cdot t$ by $s \cdot t \lor z$

Prop 7.4 in Galatos, Tsinakis (2005) Generalized MV-algebras
Checking identities in BL-algebras

To decide propositional basic logic with an SMT-solver requires the following result of Agliano Montagna 2003 (see also Aguzzoli and Bova 2010).

**Theorem**

Let \( A_n = \bigoplus_{i=0}^{n}[0, 1] \) be the ordinal sum of \( n + 1 \) unit-interval MV-algebras, and let \( \mathcal{V}_n \) be the variety generated by all \( n \)-generated BL-algebras. Then \( \mathcal{V}_n = \text{HSP}(A_n) \), hence an \( n \)-variable BL-identity holds in \( A_n \) if and only if it holds in all BL-algebras.

By constructing the algebra \( A_n \) of the above result within the SMT language, one obtains an effective means of checking \( n \)-variable BL-identities.
Checking identities in BL-algebras

The universe for $A_n$ is taken to be the interval $[0, n + 1]$

The definition of fusion and implication are

$$x \cdot y = \begin{cases} 
\max(x + y - 1 - \lfloor y \rfloor, \lfloor x \rfloor) & \text{if } \lfloor x \rfloor = \lfloor y \rfloor \\
\min(x, y) & \text{otherwise}
\end{cases}$$

$$x \rightarrow y = \begin{cases} 
n + 1 & \text{if } x \leq y \\
y & \text{if } \lfloor y \rfloor < \lfloor x \rfloor \\
\min(1 + y - x + \lfloor x \rfloor, 1 + \lfloor y \rfloor) & \text{otherwise}
\end{cases}$$

A straightforward SMT-LIB2 implementation of these operations uses $n + 1$ cases, so the formula does become long even for small values of $n$

Below we give the implementations for $n = 1$ and $n = 2$, which can be used to check 1-variable and 2-variable BL-identities
Checking identities in BL-algebras

\( n = 1: \)
\[
(\text{define-fun cdot } ((x \ Real) \ (y \ Real)) \ Real \ (\text{ite (and (< x 1) (< y 1)) (vee (- (+ x y) 1) 0) (ite (and (>= x 1) (>= y 1)) (vee (- (+ x y) 2) 1) (wedge x y))}))
\]
\[
(\text{define-fun to } ((x \ Real) \ (y \ Real)) \ Real \ (\text{ite (<= x y) 2 (ite (and (>= x 1) (< y 1)) y (wedge 1 (- (+ 1 y) x)))))})
\]

\( n = 2: \)
\[
(\text{define-fun cdot } ((x \ Real) \ (y \ Real)) \ Real \ (\text{ite (and (< x 1) (< y 1)) (vee (- (+ x y) 1) 0) (ite (and (>= x 1) (< x 2) (>= y 1) (< y 2)) (vee (- (+ x y) 2) 1) (ite (and (>= x 2) (>= y 2)) (vee (- (+ x y) 3) 2) (wedge x y)))))
\]
\[
(\text{define-fun to } ((x \ Real) \ (y \ Real)) \ Real \ (\text{ite (<= x y) 3 (ite (and (< x 1) (< y 1)) (+ (- 1 x) y) (ite (and (< 1 x) (< x 2) (< 1 y) (< y 2)) (+ (- 2 x) y) (ite (and (< 2 x) (< 2 y)) (+ (- 3 x) y) y)))))})
\]
Automating the translation

A Python program is used to parse a \LaTeX BL-algebra identity

A SMT-LIB2 file is generated using $\cdot$ and $\rightarrow$ of $A_n$

The python program then calls an SMT-solver with the file as input

The result is analyzed and the truth value is returned

If the identity fails, an assignment in $[0, n]$ can be obtained

Demo
Some References


Thank You

BLAST 2013, August 5-9, Chapman University, Orange, CA