Mining counterexamples for wide-signature algebras with an Isabelle server

Wesley Fussner, Boris Shminke

Laboratoire J.A. Dieudonné, CNRS, and Université Côte d'Azur, France

6 Sep 2021

What is a residuated binar

Binar (magma, groupoid) — a set with a binary operation $\cdot$
For residuation we add a lattice structure:

$$
\begin{aligned}
x \wedge y &= y \wedge x \\
x \wedge (y \wedge z) &= (x \wedge y) \wedge z \\
x \vee y &= y \vee x \\
x \vee (y \vee z) &= (x \vee y) \vee z \\
x \wedge (x \vee y) &= x \\
x \vee (x \wedge y) &= x
\end{aligned}
$$

What is a residuated binar (RB)

A binar with a lattice stucture ($x \leq y \iff x = x \wedge y$) and two residuation operations:

$$x \cdot y \leq z \iff x \leq z/y \iff y \leq x \backslash z$$

Some distributive laws hold in all RBs

$$x \cdot (y \vee z) = x \cdot y \vee x \cdot z$$

$$(x \vee y) \cdot z = x \cdot z \vee y \cdot z$$

$$x \backslash (y \wedge z) = x \backslash y \wedge x \backslash z$$

$$(x \wedge y)/z = x/z \wedge y/z$$

$$x/(y \vee z) = x/y \wedge x/z$$

$$(x \vee y) \backslash z = x \backslash z \wedge y \backslash z$$

And some don't (in general)

$$
\begin{aligned}
x \cdot (y \wedge z) &= x \cdot y \wedge x \cdot z \\
(x \wedge y) \cdot z &= x \cdot z \wedge y \cdot z \\
x \backslash (y \vee z) &= x \backslash y \vee x \backslash z \\
(x \vee y) / z &= x / z \vee y / z \\
(x \wedge y) \backslash z &= x \backslash z \vee y \backslash z \\
x / (y \wedge z) &= x / y \vee x / z
\end{aligned}
$$

But some distributivity laws can

- follow from a combination of others
- under special circumstances
- Fussner, W., Jipsen, P. Distributive laws in residuated binars. Algebra Univers. 80, 54 (2019)

Example from cited paper

If $(x \wedge y) \vee z = (x \wedge z) \vee (y \wedge z)$ (distributive lattice) then:

$$\begin{aligned} (x \vee y)/z &= x/z \vee y/z \\ (x \wedge y)\backslash z &= x\backslash z \vee y\backslash z \\ \Longrightarrow x\backslash(y \vee z) &= x\backslash y \vee x\backslash z \end{aligned}$$

Non-example

If $(x \wedge y) \vee z = (x \wedge z) \vee (y \wedge z)$ then there is a counter-example for:

$$\begin{aligned} x \cdot (y \wedge z) &= x \cdot y \wedge x \cdot z \\ (x \wedge y) \cdot z &= x \cdot z \wedge y \cdot z \\ \Longrightarrow x \backslash (y \vee z) &= x \backslash y \vee x \backslash z \end{aligned}$$

Open Problem

In a residuated binar which of the following distributivity laws follows from some combination of others:

$$x \cdot (y \wedge z) = x \cdot y \wedge x \cdot z \tag{1}$$

$$(x \wedge y) \cdot z = x \cdot z \wedge y \cdot z \tag{2}$$

$$x\backslash(y \vee z) = x\backslash y \vee x\backslash z \tag{3}$$

$$(x \vee y)/z = x/z \vee y/z \tag{4}$$

$$(x \wedge y)\backslash z = x\backslash z \vee y\backslash z \tag{5}$$

$$x/(y \wedge z) = x/y \vee x/z \tag{6}$$

Problem

- 6 non-trivial distributivity laws
- $(2^5 - 1) \times 6$ of possible implications between them
- adding: $\cdot$ commutativity/associativity, lattice modularity, involution operation, ...
- often a counter-examples exists

Why so many?

- if 1,2,3,4,5 doesn not imply 6
- then neither 1,2,3,4 implies 6
- but we don't know what is true in the beginning
- finding counter-examples for more general statements is harder

Task

- thousands of hypotheses
- counter-examples structure is important for understanding
- we want to check as many hypotheses as possible
- starting with the least general ones

How to find provable hypotheses?

- encode the hypothsis into some formal language
- give it to one's favourite counter-examples finder:
- Mace4, Paradox, Kodkod, ...
- wait for a couple of minutes and repeat

How to find provable hypotheses?

- ~~give it to one's favourite counter-examples finder~~ which one?
- ~~wait for a couple of minutes~~ why hours or days?
- ~~repeat~~ but we have thousands of candidates to check

How to find provable hypotheses: Isabelle Platform

- uses a relatively simple language for encoding theories
- provides an interface (through Kodkod) to SMT solvers
- Isabelle server runs solving tasks *in parallel*

Isabelle

- is overall great but
- is written in StandardML and Scala (I used Python for generating theory files)
- it's server has only TCP API (not even HTTP!)

Solution

- write a Python client to Isabelle server
- write scrips for generating and processing Isabelle theory files
- parse Isabelle server log to produce `tikz` representation of lattice reducts of counter-examples
- come up with new hypotheses to prove

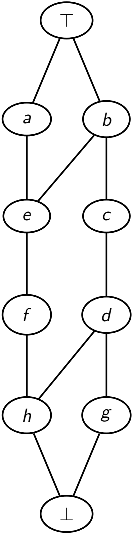
Isabelle theory file generated by Python script

```
theory T88
imports Main
begin
lemma "(
(\<forall> x::nat. \<forall> y::nat. meet(x, y) = meet(y, x)) &
(\<forall> x::nat. \<forall> y::nat. join(x, y) = join(y, x)) &
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. meet(x, meet(y, z)) = meet(meet(x, y), z)) &
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. join(x, join(y, z)) = join(join(x, y), z)) &
(\<forall> x::nat. \<forall> y::nat. meet(x, join(x, y)) = x) &
(\<forall> x::nat. \<forall> y::nat. join(x, meet(x, y)) = x) &
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. mult(x, join(y, z)) = join(mult(x, y), mult(x, z)))
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. mult(join(x, y), z) = join(mult(x, z), mult(y, z)))
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. meet(x, over(join(mult(x, y), z), y)) = x) &
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. meet(y, undr(x, join(mult(x, y), z))) = y) &
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. join(mult(over(x, y), y), x) = x) &
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. join(mult(y, undr(y, x)), x) = x) &
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. mult(x, meet(y, z)) = meet(mult(x, y), mult(x, z)))
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. mult(meet(x, y), z) = meet(mult(x, z), mult(y, z)))
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. over(join(x, y), z) = join(over(x, z), over(y, z)))
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. undr(meet(x, y), z) = join(undr(x, z), undr(y, z)))
(\<forall> x::nat. \<forall> y::nat. invo(join(x, y)) = meet(invo(x), invo(y))) &
(\<forall> x::nat. \<forall> y::nat. invo(meet(x, y)) = join(invo(x), invo(y))) &
(\<forall> x::nat. invo(invo(x)) = x)
) \<longrightarrow>
(\<forall> x::nat. \<forall> y::nat. \<forall> z::nat. undr(x, join(y, z)) = join(undr(x, y), undr(x, z)))
"
nitpick[card nat=10,timeout=86400]
oops
end
```

Isabelle server response


```
155
NOTE {"percentage":100,
"task":"1efed98a-801b-4bc8-9ea1-50b38d1d966d","message":
"theory Draft.T92 100%","kind":"writeln","session":"",
"theory":"Draft.T92"}
215033
FINISHED {"ok":true,"errors":[],"nodes":[{"messages"
:[{"kind":"writeln","message":"Nitpicking formula...",
"pos":{"line":26,"offset":1347,"end_offset":1354,"file":
"/workdir/boris/projects/residuated-binars/residuated_binar
}},{"kind":"writeln","message":
"Warning: The conjecture either trivially holds for the giv
,"pos":{"line":26,"offset":1347,"end_offset":1354,"file":
"/workdir/boris/projects/residuated-binars/residuated_binar
}},{"kind":"writeln","message":"Nitpick found a potentially
Free variables:\n    invo =\n      (\\<lambda>x. _)\n
(0 := 7, 1 := 6, 2 := 5, 3 := 4, 4 := 3, 5 := 2, 6 := 1, 7
join =\n      (\\<lambda>x. _)\n      ((0, 0) := 0, (0, 1)
```

$\cdot$	$\top$	a	b	c	d	e	f	g	h	$\bot$
$\top$	$\top$	g	$\top$	$\top$	$\top$	g	g	a	g	$\bot$
a	a	$\bot$	a	a	a	$\bot$	$\bot$	a	$\bot$	$\bot$
b	$\top$	g	$\top$	$\top$	$\top$	g	g	a	g	$\bot$
c	d	g	d	d	d	g	g	h	g	$\bot$
d	d	g	d	d	d	g	g	h	g	$\bot$
e	a	$\bot$	a	a	a	$\bot$	$\bot$	a	$\bot$	$\bot$
f	a	$\bot$	a	a	a	$\bot$	$\bot$	a	$\bot$	$\bot$
g	g	g	g	g	g	g	g	$\bot$	g	$\bot$
h	h	$\bot$	h	h	h	$\bot$	$\bot$	h	$\bot$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$


⊤
a
b
e
c
f
d
h
g
⊥

Results

In a residuated binar (with or without involution), none of the following distributivity laws follows from any combination of others:

$$x \cdot (y \wedge z) = x \cdot y \wedge x \cdot z$$
$$(x \wedge y) \cdot z = x \cdot z \wedge y \cdot z$$
$$x \backslash (y \vee z) = x \backslash y \vee x \backslash z$$
$$(x \vee y)/z = x/z \vee y/z$$
$$(x \wedge y) \backslash z = x \backslash z \vee y \backslash z$$
$$x/(y \wedge z) = x/y \vee x/z$$

Results

- all examples in the general case are non-modular
- for modular case Isabelle fails to find counter-examples of size up to 14 for some assumptions
- results from Fussner&Jipsen paper might be generalisable to the modular case

Was it easy?

- running on three Linux machines, the largest having 180 CPU cores (INTEL® XEON® Gold 6254 3.10GHz) and 832 GB of RAM
- about two weeks of wall-time computations
- filing a kernel bug report to one of the servers' sellers
- the largest model is of cardinality 10

Conclusions

- sometimes using newer software helps solving open problems in mathematics
- collaboration of mathematicians and computer scientists might be fruitful
- ITPs are not only for formalizations

Thank you for your attention!

Discussion time!