a set solver for finite set relation algebra

maxi cristiá

universidad nacional de rosario – argentina

gianfranco rossi

università degli studi di parma – italy

RAMiCS 2018 october 2018 groningen - netherlands

motivation – automated proofs – boolean algebra

single axiom C1 for boolean algebra in the sheffer stroke

$$(A \uparrow ((B \uparrow A) \uparrow A)) \uparrow (B \uparrow (A \uparrow C)) = B$$

motivation – automated proofs – boolean algebra

single axiom C1 for boolean algebra in the sheffer stroke

$$(A \uparrow ((B \uparrow A) \uparrow A)) \uparrow (B \uparrow (A \uparrow C)) = B$$

it is a theorem in set theory and set RA

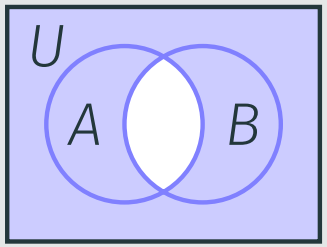
motivation – automated proofs – boolean algebra

single axiom C1 for boolean algebra in the sheffer stroke

$$(A \uparrow ((B \uparrow A) \uparrow A)) \uparrow (B \uparrow (A \uparrow C)) = B$$

it is a theorem in set theory and set RA

$$U : Set \qquad A, B, C \subseteq U$$
$$A \uparrow_U B = U \setminus (A \cap B)$$



$$A, B, C \subseteq U$$
$$U \setminus (U \setminus (A \cap (U \setminus ((U \setminus (B \cap A)) \cap A))) \cap (U \setminus (B \cap (U \setminus (A \cap C))))) = B$$

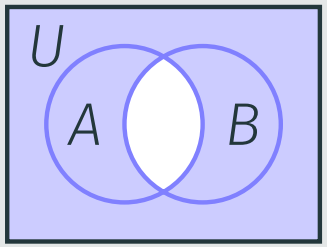
motivation – automated proofs – boolean algebra

single axiom C1 for boolean algebra in the sheffer stroke

$$(A \uparrow ((B \uparrow A) \uparrow A)) \uparrow (B \uparrow (A \uparrow C)) = B$$

it is a theorem in set theory and set RA

$$U : Set \qquad A, B, C \subseteq U$$
$$A \uparrow_U B = U \setminus (A \cap B)$$



$$A, B, C \subseteq U$$
$$U \setminus (U \setminus (A \cap (U \setminus ((U \setminus (B \cap A)) \cap A))) \cap (U \setminus (B \cap (U \setminus (A \cap C))))) = B$$

manual proof is error prone

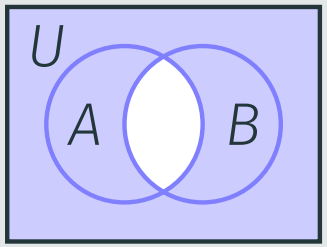
motivation – automated proofs – boolean algebra

single axiom C1 for boolean algebra in the sheffer stroke

$$(A \uparrow ((B \uparrow A) \uparrow A)) \uparrow (B \uparrow (A \uparrow C)) = B$$

it is a theorem in set theory and set RA

$$U : Set \qquad A, B, C \subseteq U$$
$$A \uparrow_U B = U \setminus (A \cap B)$$



$$A, B, C \subseteq U$$
$$U \setminus (U \setminus (A \cap (U \setminus ((U \setminus (B \cap A)) \cap A))) \cap (U \setminus (B \cap (U \setminus (A \cap C))))) = B$$

manual proof is error prone

automated proof?

motivation – automated proofs – set RA

the cycle law (or De Morgan's theorem K)

$$X ; Y \cdot Z = 0 \iff \breve{X} ; Z \cdot Y = 0$$

motivation – automated proofs – set RA

the cycle law (or De Morgan's theorem K)

$$X ; Y \cdot Z = 0 \iff \breve{X} ; Z \cdot Y = 0$$

can also be stated as a set theory theorem

$$X ; Y \cap Z = \emptyset \iff \breve{X} ; Z \cap Y = \emptyset$$

provided composition and converse are defined

motivation – automated proofs – set RA

the cycle law (or De Morgan's theorem K)

$$X ; Y \cdot Z = 0 \iff \breve{X} ; Z \cdot Y = 0$$

can also be stated as a set theory theorem

$$X ; Y \cap Z = \emptyset \iff \breve{X} ; Z \cap Y = \emptyset$$

provided composition and converse are defined

automated proof?

motivation – automated proofs – set RA

many interesting notions can be defined as RA formulas

R is reflexive	iff	$1' \subseteq R$
R is transitive	iff	$R ; R \subseteq R$
R is idempotent	iff	$R ; R = R$

motivation – automated proofs – set RA

many interesting notions can be defined as RA formulas

R is reflexive	iff	$1' \subseteq R$
R is transitive	iff	$R ; R \subseteq R$
R is idempotent	iff	$R ; R = R$

so the following holds

if R is reflexive and transitive, then is idempotent

$1' \subseteq R \wedge R ; R \subseteq R \implies R ; R = R$

motivation – automated proofs – set RA

many interesting notions can be defined as RA formulas

R is reflexive	iff	$1' \subseteq R$
R is transitive	iff	$R ; R \subseteq R$
R is idempotent	iff	$R ; R = R$

so the following holds

if R is reflexive and transitive, then is idempotent

$1' \subseteq R \wedge R ; R \subseteq R \implies R ; R = R$

automated proof?

motivation – finding counterexamples

is the following is a theorem?

$$X \,;\, Y \cdot \overline{Z} = X \,;\, Y \cdot \overline{X \,;\, Z}$$

motivation – finding counterexamples

is the following is a theorem?

$$X ; Y \cdot \overline{Z} = X ; Y \cdot \overline{X ; Z}$$

no, it isn't

why not?
counterexamples may help to find the missing condition

$A = \{x_1, x_2\}, B = \{x_3\}, x_1 \neq x_2,$ set RA over $A \times B$
$X = \{(x_1, x_2)\}, Y = \{(x_2, x_3)\}, Z = \{(x_2, x_3)\}$ $X \not\subseteq 1'$

motivation – finding counterexamples

is the following is a theorem?

$$X ; Y \cdot \overline{Z} = X ; Y \cdot \overline{X ; Z}$$

no, it isn't

why not?
counterexamples may help to find the missing condition

$A = \{x_1, x_2\}, B = \{x_3\}, x_1 \neq x_2,$ set RA over $A \times B$
$X = \{(x_1, x_2)\}, Y = \{(x_2, x_3)\}, Z = \{(x_2, x_3)\}$ $X \not\subseteq 1'$

can we get counterexamples automatically?

motivation – computing relations

if $A = \{a, b, c\}$, compute all transitive, irreflexive relations

motivation – computing relations

if $A = \{a, b, c\}$, compute all transitive, irreflexive relations

$$A = \{a, b, c\} \land R \subseteq A \times A \land R \,;\, R \subseteq R \land R \parallel \mathrm{id}(A)$$

motivation – computing relations

if $A = \{a, b, c\}$, compute all transitive, irreflexive relations

$$A = \{a, b, c\} \land R \subseteq A \times A \land R \,;\, R \subseteq R \land R \parallel \mathrm{id}(A)$$

$$\{(a, b)\} \quad \{(b, c), (a, c)\} \quad \{(c, b), (b, a), (c, a)\} \quad \ldots$$

motivation – computing relations

if $A = \{a, b, c\}$, compute all transitive, irreflexive relations

$$A = \{a, b, c\} \land R \subseteq A \times A \land R \,;\, R \subseteq R \land R \parallel \mathrm{id}(A)$$

$$\{(a, b)\} \quad \{(b, c), (a, c)\} \quad \{(c, b), (b, a), (c, a)\} \quad \ldots$$

this may be easy to automate… but…

motivation – computing relations

if $A = \{a, b, c\}$, compute all transitive, irreflexive relations

$$A = \{a, b, c\} \land R \subseteq A \times A \land R \,;\, R \subseteq R \land R \parallel \mathrm{id}(A)$$

$$\{(a, b)\} \quad \{(b, c), (a, c)\} \quad \{(c, b), (b, a), (c, a)\} \quad \ldots$$

this may be easy to automate... but...

all in the same tool?

{*log*}

{*log*} can do all of this for *finite, unbounded relations*

{*log*} is an automated theorem prover

- satisfiability solver
- set theory $\rightarrow$ boolean algebra, set relation algebra

{*log*} is a counterexample generator

{*log*} is a set-based, constraint-based programming language

{*log*} – before this paper

sets and binary relations

$\emptyset$	$\rightarrow$ the empty set
$\{elem \sqcup set\}$	$\rightarrow$ the set $\{elem\} \cup set$
variable	$\rightarrow$ any finite, unbounded set

binary relations are sets of ordered pairs

{*log*} – before this paper

sets and binary relations

$\emptyset$	$\rightarrow$ the empty set
$\{elem \sqcup set\}$	$\rightarrow$ the set $\{elem\} \cup set$
variable	$\rightarrow$ any finite, unbounded set

binary relations are sets of ordered pairs

operators

$=$ $\quad \cup \quad \in \quad \| \quad ; \quad \mathrm{dom} \quad \mathrm{ran} \quad \mathrm{pfun} \quad \wedge \quad \vee \quad \neg$

$\neg A = B \iff A \neq B \iff x \in A \wedge x \notin B \vee x \notin A \wedge x \in B$ $\quad A, B$ sets

{*log*} – before this paper

sets and binary relations

$\emptyset$	$\rightarrow$ the empty set
$\{elem \sqcup set\}$	$\rightarrow$ the set $\{elem\} \cup set$
variable	$\rightarrow$ any finite, unbounded set

binary relations are sets of ordered pairs

operators

$=$ $\quad \cup \quad \in \quad \| \quad ; \quad \mathrm{dom} \quad \mathrm{ran} \quad \mathrm{pfun} \quad \wedge \quad \vee \quad \neg$

$\neg A = B \iff A \neq B \iff x \in A \wedge x \notin B \vee x \notin A \wedge x \in B$ $\quad A, B$ sets

operators are turned into predicates

$$R ; S = T \iff comp(R, S, T)$$

{*log*} – before this paper

sets and binary relations

$\emptyset$	$\rightarrow$ the empty set
$\{elem \sqcup set\}$	$\rightarrow$ the set $\{elem\} \cup set$
variable	$\rightarrow$ any finite, unbounded set

binary relations are sets of ordered pairs

operators

$=$ $\quad \cup \quad \in \quad \| \quad ; \quad \mathrm{dom} \quad \mathrm{ran} \quad \mathrm{pfun} \quad \wedge \quad \vee \quad \neg$

$\neg A = B \iff A \neq B \iff x \in A \wedge x \notin B \vee x \notin A \wedge x \in B$ $\quad A, B$ sets

operators are turned into predicates

$$R ; S = T \iff comp(R, S, T)$$

many more operators can be introduced as formulas

counterexamples – solutions

if the formula is satisfiable,

$\{log\}$ returns a finite representation of all its solutions

for	$comp(\{(w,x) \sqcup R\}, \{(y,z) \sqcup S\}, T)$	the answer is
	$y = x, T = \{(w,z) \sqcup N_1\}$	
	$un(N_3, N_2, N_1), comp(\{(w,x)\}, S, N_3), comp(R, \{(x,z) \sqcup S\}, N_2)$	
$\vee$	$x \neq y, un(N_2, N_1, T), comp(\{(w,x)\}, S, N_2), comp(R, \{(y,z) \sqcup S\}, N_1)$	

N_i new variables, existentially quantified

each disjunct is satisfied with $\emptyset$

$$y = x \wedge R = S = N_i = \emptyset$$

$$\Longrightarrow \{(w,x)\} ; \{(y,z)\} = \{(w,x)\} ; \{(x,z)\} = \{(w,z)\} = T$$

automated proofs

if the formula is unsatisfiable, $\{log\}$ returns *false*

useful to prove theorems

theorem (in mathematics)

$$\operatorname{ran} R \parallel \operatorname{dom} S \implies R ; S = \emptyset$$

the negation of the theorem (in $\{log\}$)

$$ran(R, A) \wedge dom(S, B) \wedge A \parallel B \wedge ncomp(R, S, \emptyset)$$

expressing set RA

before this paper, **could we express set RA in $\{log\}$?**

expressing set RA

before this paper, **could we express set RA in $\{log\}$?**

no, we couldn't

we had union, intersection and composition

we could define 1, $1'$ and complement

$$A \times B = A \times \{n\} \,;\, \{n\} \times B \qquad R = A \times \{n\} \iff \mathbf{dom}\, R = A \wedge \mathbf{ran}\, R \subseteq \{n\}$$

$$\overline{R} = S \iff un(R, S, A \times A) \wedge R \parallel S \qquad \text{in a set RA over } A$$

but we couldn't define converse

and the definitions for 1 and $1'$ are too inefficient

expressing set RA

then, in this paper

- identity becomes a primitive operator
- cartesian product becomes a new set term
- converse becomes a new primitive operator
- dom, ran and pfun *can* become defined operators
  - for efficiency reasons they are kept as primitive

the language is kept to a minimum

$\{log\}$'s main features remain the same

{*log*} internals

{*log*} is a nondeterministic rewriting system

→ it nondeterministically picks a literal

→ applies a rewrite rule to it

→ which returns one or more formulas

→ until a fixpoint is reached

{*log*} internals

{*log*} is a nondeterministic rewriting system

→ it nondeterministically picks a literal

→ applies a rewrite rule to it

→ which returns one or more formulas

→ until a fixpoint is reached

an example of a rewrite rule – **set unification**

$$\{t \sqcup A\} = \{u \sqcup B\} \rightarrow$$

recall: $\{t \sqcup A\}$ is $\{t\} \cup A$

$$t = u \wedge A = \{u \sqcup B\}$$

u, t, A, B may be variables

$$\vee\, t = u \wedge B = \{t \sqcup A\}$$

$$\vee\, t = u \wedge A = B$$

$$\vee\, A = \{u \sqcup N\} \wedge B = \{t \sqcup N\}$$

N new variable

dealing with cartesian products

when $A \times B$ is a set term we need to solve equations such as

$X \times Y = \{z \sqcup Z\}$ z, X, Y, Z may be variables

dealing with cartesian products

when $A \times B$ is a set term we need to solve equations such as

$$X \times Y = \{z \sqcup Z\} \to$$
$$z = (n_1, n_2)$$

z, X, Y, Z may be variables

n_i new variables

dealing with cartesian products

when $A \times B$ is a set term we need to solve equations such as

$$X \times Y = \{z \sqcup Z\} \to$$
$$z = (n_1, n_2)$$
$$\wedge\, X = \{n_1 \sqcup N_1\} \wedge Y = \{n_2 \sqcup N_2\}$$

z, X, Y, Z may be variables

n_i new variables

N_i new variables

dealing with cartesian products

when $A \times B$ is a set term we need to solve equations such as

$$
\begin{aligned}
&X \times Y = \{z \sqcup Z\} \rightarrow && z, X, Y, Z \text{ may be variables} \\
&\quad z = (n_1, n_2) && n_i \text{ new variables} \\
&\quad \wedge\, X = \{n_1 \sqcup N_1\} \wedge Y = \{n_2 \sqcup N_2\} && N_i \text{ new variables} \\
&\quad \wedge\, un(\{n_1\} \times N_2, N_1 \times \{n_2 \sqcup N_2\}, Z)
\end{aligned}
$$

dealing with cartesian products

when $A \times B$ is a set term we need to solve equations such as

$$
\begin{aligned}
&X \times Y = \{z \sqcup Z\} \rightarrow && z, X, Y, Z \text{ may be variables} \\
&\quad z = (n_1, n_2) && n_i \text{ new variables} \\
&\quad \wedge\, X = \{n_1 \sqcup N_1\} \wedge Y = \{n_2 \sqcup N_2\} && N_i \text{ new variables} \\
&\quad \wedge\, un(\{n_1\} \times N_2, N_1 \times \{n_2 \sqcup N_2\}, Z)
\end{aligned}
$$

we need rewrite rules for union

dealing with cartesian products

when $A \times B$ is a set term we need to solve equations such as

$$X \times Y = \{z \sqcup Z\} \rightarrow$$
$$z = (n_1, n_2)$$
$$\wedge\, X = \{n_1 \sqcup N_1\} \wedge Y = \{n_2 \sqcup N_2\}$$
$$\wedge\, un(\{n_1\} \times N_2, N_1 \times \{n_2 \sqcup N_2\}, Z)$$

z, X, Y, Z may be variables

n_i new variables

N_i new variables

we need rewrite rules for union
for example:

$$un(\{x \sqcup X\} \times \{y \sqcup Y\}, B, C) \rightarrow$$
$$un(\{(x, y) \sqcup N_1\}, B, C) \wedge un(\{x\} \times Y, X \times \{y \sqcup Y\}, N_1)$$

dealing with cartesian products

when $A \times B$ is a set term we need to solve equations such as

$$X \times Y = \{z \sqcup Z\} \rightarrow$$
$$z = (n_1, n_2)$$
$$\wedge\, X = \{n_1 \sqcup N_1\} \wedge Y = \{n_2 \sqcup N_2\}$$
$$\wedge\, un(\{n_1\} \times N_2, N_1 \times \{n_2 \sqcup N_2\}, Z)$$

z, X, Y, Z may be variables

n_i new variables

N_i new variables

we need rewrite rules for union
for example:

$$un(\{x \sqcup X\} \times \{y \sqcup Y\}, B, C) \rightarrow$$
$$un(\{(x, y) \sqcup N_1\}, B, C) \wedge un(\{x\} \times Y, X \times \{y \sqcup Y\}, N_1)$$

seems to be a loop...

when to stop?

$A \times B$ is a *variable product* if A or B are variables

union rewriting stops when all arguments are variables or variable products

when to stop?

$A \times B$ is a *variable product* if A or B are variables

union rewriting stops when all arguments are variables or variable products

$un(\{x\} \times Y, X \times \{y \sqcup Y\}, N_1)$

isn't further rewritten if X and Y are variables

it is satisfied with $X = Y = N_1 = \emptyset$ N_1 is a variable

when to stop?

$A \times B$ is a *variable product* if A or B are variables

union rewriting stops when all arguments are variables or variable products

$un(\{x\} \times Y, X \times \{y \sqcup Y\}, N_1)$

isn't further rewritten if X and Y are variables

it is satisfied with $X = Y = N_1 = \emptyset$ N_1 is a variable

$un(\{(x, y) \sqcup N_1\}, B, C)$

if B and C are variables N_1 is a variable

is further rewritten to $C = \{(x, y) \sqcup N_2\} \wedge un(N_1, B, N_2)$

but $un(N_1, B, N_2)$ is not because all are variables

the converse operator

$$inv(R, S) \iff \breve{R} = S$$

the easy rewrite rules

$inv(R, S)$ isn't rewritten if R and S are variables

$$inv(R, \emptyset) \to R = \emptyset$$

$$inv(R, \{(y, x) \sqcup S\}) \to R = \{(x, y) \sqcup N\} \wedge inv(N, S)$$

$$inv(X \times Y, S) \to S = Y \times X$$

the converse operator

the simplified version of the complex rewrite rule

R is a variable

$$
\begin{aligned}
&inv(\{(x,y) \sqcup R\}, \{(a,b) \sqcup R\}) \rightarrow \\
&\quad (y,x) = (a,b) \wedge inv(R,R) \\
&\quad \vee (y,x) \neq (a,b) \wedge (x,y) = (a,b) \\
&\qquad \wedge R = \{(y,x) \sqcup N\} \wedge inv(N,N) \\
&\quad \vee (y,x) \neq (a,b) \wedge (x,y) \neq (a,b) \\
&\qquad \wedge R = \{(x,y),(y,x) \sqcup N\} \wedge inv(N, \{(a,b) \sqcup N\})
\end{aligned}
$$

expressiveness

therefore, we have

finite, unbounded sets of ordered pairs

cartesian products

equality, union, complement,
composition, converse, identity

conjunction, disjunction, negation

hence, we can express

heterogeneous finite set relation algebras

solved form

when rewriting finishes, the returned answer can be

false or

a disjunction of formulas in *solved form*

each formula is a solution or counterexample

solved form

when rewriting finishes, the returned answer can be

false or

a disjunction of formulas in *solved form*

each formula is a solution or counterexample

literals in solved form X, X_i are variables or variable products

$V = t$	$un(X_1, X_2, X_3)$	$X \neq t$ [X, t not sets]
$t \notin X$	$id(X_1, X_2)$	$comp(X_1, t, s)$ [s variable or $\emptyset$]
$X_1 \parallel X_2$	$inv(X_1, X_2)$	$comp(t, X_1, s)$ [s variable or $\emptyset$]

satisfiability of solved form

Theorem

Any conjunction of literals in solved form is satisfiable w.r.t. the intended interpretation structure.

Proof

Substitute set variables by the empty set.

soundness

Theorem

Let Φ be a formula and let $\{\phi_i\}_{i=1}^{n}$ be the collection of formulas returned by $\{log\}$. Then $\bigvee_{i=1}^{n} \phi_i$ is equisatisfiable to Φ.

Proof

For each rewrite rule we prove that a solution of the l.h.s. is a solution of the r.h.s, and vice-versa.

termination

termination can't be guaranteed given that the class of finite set relation algebras has been proved to be not decidable

then, for some formulas, {*log*} may

- return an infinite number of formulas in solved form
  - the formula is satisfiable but you won't get all its solutions

termination

termination can't be guaranteed given that the class of finite set relation algebras has been proved to be not decidable

then, for some formulas, $\{log\}$ may

- return an infinite number of formulas in solved form
  the formula is satisfiable but you won't get all its solutions

  or
- not return anything, after returning zero or more formulas
  you can't tell anything about the formula if nothing is returned

termination

termination can't be guaranteed given that the class of finite set relation algebras has been proved to be not decidable

then, for some formulas, {*log*} may

- return an infinite number of formulas in solved form
  the formula is satisfiable but you won't get all its solutions

  or
- not return anything, after returning zero or more formulas
  you can't tell anything about the formula if nothing is returned

is non termination a problem in practice?

the practical side

{*log*} can do all the motivating examples, and more

PROBLEM SET	PROBLEMS	SOLVED	PERCENTAGE	AVG TIME
TPTP.BOO	142	138	97%	0.04s
TPTP.REL	72	70	97%	0.07s
TPTP.SET	735	702	95%	0.03s
ZMT	285	285	100%	0.03s
RA	122	120	98%	0.04s
SUMMARY	**1,356**	**1,317**	**97%**	**0.03s**

non termination doesn't seem to be a problem in practice

{*log*}'s latest version is faster than paper's

concluding remarks

$\{log\}$ is at least as expressive as finite set RA

it performs well in practice

automated theorem prover

counterexample generator

however, termination can't be guaranteed

non termination *only* affects formulas containing

$\mathcal{E}_1(R)\,;\mathcal{E} = \mathcal{E}_2(R)$ or $\mathcal{E}\,;\mathcal{E}_1(S) = \mathcal{E}_2(S)$

but not all of them

R is a function $\breve{S}$ is a function

future work

{*log*}-based ITP for non decidable formulas

restricted intensional sets

powerset, generalized union

better integer reasoning, to support reasoning over lists