Using Gentzen system techniques with existing automated theorem provers and rewriting tools

Peter Jipsen

Chapman Univ, Orange, CA

March 6, 2008

Outline

- Background
- Gentzen systems
- Term rewriting systems
- Maude
- Theorem provers
- Prover9

Outline

- Background
- Gentzen systems
- Term rewriting systems
- Maude
- Theorem provers
- Prover9

Outline

- Background
- Gentzen systems
- Term rewriting systems
- Maude
- Theorem provers
- Prover9

Outline

- Background
- Gentzen systems
- Term rewriting systems
- Maude
- Theorem provers
- Prover9

Outline

- Background
- Gentzen systems
- Term rewriting systems
- Maude
- Theorem provers
- Prover9

Outline

- Background
- Gentzen systems
- Term rewriting systems
- Maude
- Theorem provers
- Prover9

Background

In the late '80s I learned about term-rewriting

from the algebraic perspective:

normal form, Church-Rosser property (confluence), termination, critical pairs, Knuth-Bendix completion ...

Applies to (free) monoids, groups, boolean algebras, (some) finitely presented monoids, groups, ...

Used in algebraic specification languages, in optimization of functional programs, in equational theorem provers, ...

Background

In the late '90s I learned about sequents and Gentzen systems

from the logical perspective:

Developed for classical logic (LK), intuitionistic logic (LJ), linear logic (MALL, IMALL), ...

For substructural logics in general

Gentzen systems provide decision procedures for many logics

Related to natural deduction, tableau methods, resolution theorem provers, ...

Aims of this talk

1. Show how a standard term-rewriting system (e.g. Maude2) can be used to implement a Gentzen calculus

$\Longrightarrow$ program to decide for RL, CRL, IRL, ICRL, InRL, ... equations

extend this approach to Hypersequent calculi

2. Adding sequent rules to an automated theorem prover (e.g. Prover9) can improve its performance

Term-rewriting

Given a term t and an equation $r = s$, a *rewrite step* consists of

matching r to a subterm of t

i.e. find a substitution σ such that $\sigma(r)$ is a subterm of t

then replace this subterm with $\sigma(s)$ to get a new term t'.

This is denoted by $t \longrightarrow t'$ or $t \xrightarrow{r=s} t'$

Note that if $r = s$ is an identity in a theory, then $t = t'$ holds in this theory.

E.g. $x = (x \vee y) \wedge x$ is used to rewrite $x \vee x \longrightarrow ((x \vee y) \wedge x) \vee x$

then $(y \wedge x) \vee x = x$ is used to rewrite $((x \vee y) \wedge x) \vee x \longrightarrow x$

Term-rewriting tools

A *term-rewriting tool* is a program that implements the process of term-rewriting for a given algebraic signature

Takes a set of *rewrite rules* (i.e. equations $r = s$) and a term t as input and applies the rules repeatedly (from left to right) and fairly to t

i.e. the tool computes the transitive closure $\longrightarrow^*$ of $t \longrightarrow t'$

This can be a very efficient way of doing calculations on a computer

The set of rewrite rules is the "program", the rewriting tool is the *interpreter* that runs the program

The process is *easily* parallelized and distributed over many processors

Dozens of term-rewriting systems have been developed, e.g. for prototyping algebraic specifications, implementing model checkers, ...

Maude

Maude is an order-sorted conditional term-rewriting tool with search capabilities

Developed by SRI International and University of Illinois at Champaign Urbana

Open source (GPL, free download), well documented (450 page manual)

Efficient and flexible, allows the use of convenient syntax

```
fmod DLAT-TERMS is
   sorts Term .
   ops _V_  _^_ : Term Term -> Term [assoc comm] .
   eq  (x V y) ^ x = x .
   eq  (x ^ y) V x = x .
   eq  x V (y ^ z) = (x V y) ^ (x V z) .
endfm
```

Gentzen sequent calculi

formulas ϕ, ψ, θ, sequences of formulas Γ, Δ, Σ, sequents $\Gamma \Rightarrow \Sigma$

sequent rules e.g. $\dfrac{\Gamma, \phi, \Delta \Rightarrow \Sigma \quad \Gamma, \psi, \Delta \Rightarrow \Sigma}{\Gamma, \phi \vee \psi, \Delta \Rightarrow \Sigma}\ (\vee\text{-l})$

Main idea is to view sequent rules as rewrite rules (upwards):

$(\Gamma, \phi \vee \psi, \Delta \Rightarrow \Sigma) \quad \rightarrow \quad (\Gamma, \phi, \Delta \Rightarrow \Sigma \quad \Gamma, \psi, \Delta \Rightarrow \Sigma)$

Prefer to view this algebraically (not essential):

$u(x \vee y)v \leq z \quad \rightarrow \quad (uxv \leq z \quad uyv \leq z)$

Here u, v, x, y, z can be instantiated with arbitrary terms

uv is an associative operation symbol

Deciding equations in RL

RL is short for *Residuated Lattice* (lattice with monoid operation + residuals)

Suppose someone has a huge RL-term t and asks if $t = 1$ is true in RL

What should we do?

- Implement the cut-free sequent calculus for RL. But how?
- On my homepage is a JavaScript implementation. But is it correct? Efficient?
- Rather use the sequent rules as rewrite rules and write a short Maude module that is easy to check
- Same approach works for CRL, IRL, InCRL, FL, FL_e, FL_{ew}, ...

Deciding equations in RL

RL is short for *Residuated Lattice* (lattice with monoid operation + residuals)

Suppose someone has a huge RL-term t and asks if $t = 1$ is true in RL

What should we do?

- Implement the cut-free sequent calculus for RL. But how?
- On my homepage is a JavaScript implementation. But is it correct? Efficient?
- Rather use the sequent rules as rewrite rules and write a short Maude module that is easy to check
- Same approach works for CRL, IRL, InCRL, FL, FL_e, FL_{ew}, ...

Deciding equations in RL

RL is short for *Residuated Lattice* (lattice with monoid operation + residuals)

Suppose someone has a huge RL-term t and asks if $t = 1$ is true in RL

What should we do?

- Implement the cut-free sequent calculus for RL. But how?
- On my homepage is a JavaScript implementation. But is it correct? Efficient?
- Rather use the sequent rules as rewrite rules and write a short Maude module that is easy to check
- Same approach works for CRL, IRL, InCRL, FL, FL_e, FL_{ew}, ...

Deciding equations in RL

RL is short for *Residuated Lattice* (lattice with monoid operation + residuals)

Suppose someone has a huge RL-term t and asks if $t = 1$ is true in RL

What should we do?

- Implement the cut-free sequent calculus for RL. But how?
- On my homepage is a JavaScript implementation. But is it correct? Efficient?
- Rather use the sequent rules as rewrite rules and write a short Maude module that is easy to check
- Same approach works for CRL, IRL, InCRL, FL, FL_e, FL_{ew}, ...

Deciding equations in RL

RL is short for *Residuated Lattice* (lattice with monoid operation + residuals)

Suppose someone has a huge RL-term t and asks if $t = 1$ is true in RL

What should we do?

- Implement the cut-free sequent calculus for RL. But how?
- On my homepage is a JavaScript implementation. But is it correct? Efficient?
- Rather use the sequent rules as rewrite rules and write a short Maude module that is easy to check
- Same approach works for CRL, IRL, InCRL, FL, FL_e, FL_{ew}, ...

Demo of Gentzen systems implemented in Maude

The files boolean-algebra.maude and RL-sequent.maude (and other files) can be found at

`http://mathcs.chapman.edu/~jipsen/maude/files`

Deciding equations in BA

BA is short for *Boolean algebras*

Suppose someone has a huge BA-term t and asks if $t = 1$ is true in BA

What should we do?

- Implement the sequent calculus for LK?
- Rather use an off-the-shelf SAT-solver (tests satisfiability of Boolean formulas)
- Highly optimized, used for hardware verification
- Same approach works for testing whether an equation holds in a finitely generated variety
- and for checking tautologies in propositional logics with a finite matrix of truth values

Deciding equations in BA

BA is short for *Boolean algebras*

Suppose someone has a huge BA-term t and asks if $t = 1$ is true in BA

What should we do?

- Implement the sequent calculus for LK?
- Rather use an off-the-shelf SAT-solver (tests satisfiability of Boolean formulas)
- Highly optimized, used for hardware verification
- Same approach works for testing whether an equation holds in a finitely generated variety
- and for checking tautologies in propositional logics with a finite matrix of truth values

Deciding equations in BA

BA is short for *Boolean algebras*

Suppose someone has a huge BA-term t and asks if $t = 1$ is true in BA

What should we do?

- Implement the sequent calculus for LK?
- Rather use an off-the-shelf SAT-solver (tests satisfiability of Boolean formulas)
- Highly optimized, used for hardware verification
- Same approach works for testing whether an equation holds in a finitely generated variety
- and for checking tautologies in propositional logics with a finite matrix of truth values

Deciding equations in BA

BA is short for *Boolean algebras*

Suppose someone has a huge BA-term t and asks if $t = 1$ is true in BA

What should we do?

- Implement the sequent calculus for LK?
- Rather use an off-the-shelf SAT-solver (tests satisfiability of Boolean formulas)
- Highly optimized, used for hardware verification
- Same approach works for testing whether an equation holds in a finitely generated variety
- and for checking tautologies in propositional logics with a finite matrix of truth values

Deciding equations in BA

BA is short for *Boolean algebras*

Suppose someone has a huge BA-term t and asks if $t = 1$ is true in BA

What should we do?

- Implement the sequent calculus for LK?
- Rather use an off-the-shelf SAT-solver (tests satisfiability of Boolean formulas)
- Highly optimized, used for hardware verification
- Same approach works for testing whether an equation holds in a finitely generated variety
- and for checking tautologies in propositional logics with a finite matrix of truth values

Deciding equations in BA

BA is short for *Boolean algebras*

Suppose someone has a huge BA-term t and asks if $t = 1$ is true in BA

What should we do?

- Implement the sequent calculus for LK?
- Rather use an off-the-shelf SAT-solver (tests satisfiability of Boolean formulas)
- Highly optimized, used for hardware verification
- Same approach works for testing whether an equation holds in a finitely generated variety
- and for checking tautologies in propositional logics with a finite matrix of truth values

Deciding equations in abelian l-groups

Ab-LG is short for *abelian lattice ordered groups*

Suppose someone has a huge Ab-LG-term t and asks if $t = 1$ is true in Ab-LG

What should we do?

- Implement the hypersequent calculus for abelian logic (Metcalfe-Olivetti-Gabbay 2004)
- Can be done with a term rewriting tool
- Alternatively use a SMT-solver (symbolic model checking modulo theories)
- Need to compare these approaches to see what works better in practise
- Same works whenever the logic has a terminating hypersequent calculus (product logic, MV-algebras, linear Heyting algebras, ...)

Deciding equations in abelian l-groups

Ab-LG is short for *abelian lattice ordered groups*

Suppose someone has a huge Ab-LG-term t and asks if $t = 1$ is true in Ab-LG

What should we do?

- Implement the hypersequent calculus for abelian logic (Metcalfe-Olivetti-Gabbay 2004)
- Can be done with a term rewriting tool
- Alternatively use a SMT-solver (symbolic model checking modulo theories)
- Need to compare these approaches to see what works better in practise
- Same works whenever the logic has a terminating hypersequent calculus (product logic, MV-algebras, linear Heyting algebras, ...)

Deciding equations in abelian l-groups

Ab-LG is short for *abelian lattice ordered groups*

Suppose someone has a huge Ab-LG-term t and asks if $t = 1$ is true in Ab-LG

What should we do?

- Implement the hypersequent calculus for abelian logic (Metcalfe-Olivetti-Gabbay 2004)
- Can be done with a term rewriting tool
- Alternatively use a SMT-solver (symbolic model checking modulo theories)
- Need to compare these approaches to see what works better in practise
- Same works whenever the logic has a terminating hypersequent calculus (product logic, MV-algebras, linear Heyting algebras, ...)

Deciding equations in abelian l-groups

Ab-LG is short for *abelian lattice ordered groups*

Suppose someone has a huge Ab-LG-term t and asks if $t = 1$ is true in Ab-LG

What should we do?

- Implement the hypersequent calculus for abelian logic (Metcalfe-Olivetti-Gabbay 2004)
- Can be done with a term rewriting tool
- Alternatively use a SMT-solver (symbolic model checking modulo theories)
- Need to compare these approaches to see what works better in practise
- Same works whenever the logic has a terminating hypersequent calculus (product logic, MV-algebras, linear Heyting algebras, ...)

Deciding equations in abelian l-groups

Ab-LG is short for *abelian lattice ordered groups*

Suppose someone has a huge Ab-LG-term t and asks if $t = 1$ is true in Ab-LG

What should we do?

- Implement the hypersequent calculus for abelian logic (Metcalfe-Olivetti-Gabbay 2004)
- Can be done with a term rewriting tool
- Alternatively use a SMT-solver (symbolic model checking modulo theories)
- Need to compare these approaches to see what works better in practise
- Same works whenever the logic has a terminating hypersequent calculus (product logic, MV-algebras, linear Heyting algebras, ...)

Deciding equations in abelian l-groups

Ab-LG is short for *abelian lattice ordered groups*

Suppose someone has a huge Ab-LG-term t and asks if $t = 1$ is true in Ab-LG

What should we do?

- Implement the hypersequent calculus for abelian logic (Metcalfe-Olivetti-Gabbay 2004)
- Can be done with a term rewriting tool
- Alternatively use a SMT-solver (symbolic model checking modulo theories)
- Need to compare these approaches to see what works better in practise
- Same works whenever the logic has a terminating hypersequent calculus (product logic, MV-algebras, linear Heyting algebras, ...)

Deciding equations in commutative CGBL

CGBL is short for *commutative generalized basic logic algebras*

= RL with $xy = yx$ and divisibility: $x \leq y \Rightarrow x = y(y\backslash x) = (x/y)y$

Suppose someone has a CGBL-term t and asks if $t = 1$ is true in CGBL

What should we do?

Theorem (J - Montagna 2008)

The varieties ICGBL and CGBL have the finite model property (FEP), hence their universal theory is decidable

- Currently there is no "reasonable" proof procedure
- Can use an automated theorem prover and finite model finder
- E.g. Prover9 and Mace4
- Same works (in principle) whenever we have FEP or FMP but not really effective

Deciding equations in commutative CGBL

CGBL is short for *commutative generalized basic logic algebras*

= RL with $xy = yx$ and divisibility: $x \leq y \Rightarrow x = y(y\backslash x) = (x/y)y$

Suppose someone has a CGBL-term t and asks if $t = 1$ is true in CGBL

What should we do?

Theorem (J - Montagna 2008)

The varieties ICGBL and CGBL have the finite model property (FEP), hence their universal theory is decidable

- Currently there is no "reasonable" proof procedure
- Can use an automated theorem prover and finite model finder
- E.g. Prover9 and Mace4
- Same works (in principle) whenever we have FEP or FMP but not really effective

Deciding equations in commutative CGBL

CGBL is short for *commutative generalized basic logic algebras*

= RL with $xy = yx$ and divisibility: $x \leq y \Rightarrow x = y(y\backslash x) = (x/y)y$

Suppose someone has a CGBL-term t and asks if $t = 1$ is true in CGBL

What should we do?

Theorem (J - Montagna 2008)

The varieties ICGBL and CGBL have the finite model property (FEP), hence their universal theory is decidable

- Currently there is no "reasonable" proof procedure
- Can use an automated theorem prover and finite model finder
- E.g. Prover9 and Mace4
- Same works (in principle) whenever we have FEP or FMP but not really effective

Deciding equations in commutative CGBL

CGBL is short for *commutative generalized basic logic algebras*

= RL with $xy = yx$ and divisibility: $x \leq y \Rightarrow x = y(y\backslash x) = (x/y)y$

Suppose someone has a CGBL-term t and asks if $t = 1$ is true in CGBL

What should we do?

Theorem (J - Montagna 2008)

The varieties ICGBL and CGBL have the finite model property (FEP), hence their universal theory is decidable

- Currently there is no "reasonable" proof procedure
- Can use an automated theorem prover and finite model finder
- E.g. Prover9 and Mace4
- Same works (in principle) whenever we have FEP or FMP but not really effective

Deciding equations in commutative CGBL

CGBL is short for *commutative generalized basic logic algebras*

= RL with $xy = yx$ and divisibility: $x \leq y \Rightarrow x = y(y\backslash x) = (x/y)y$

Suppose someone has a CGBL-term t and asks if $t = 1$ is true in CGBL

What should we do?

Theorem (J - Montagna 2008)

The varieties ICGBL and CGBL have the finite model property (FEP), hence their universal theory is decidable

- Currently there is no "reasonable" proof procedure
- Can use an automated theorem prover and finite model finder
- E.g. Prover9 and Mace4
- Same works (in principle) whenever we have FEP or FMP but not really effective

Deciding equations in commutative CGBL

CGBL is short for *commutative generalized basic logic algebras*

= RL with $xy = yx$ and divisibility: $x \leq y \Rightarrow x = y(y\backslash x) = (x/y)y$

Suppose someone has a CGBL-term t and asks if $t = 1$ is true in CGBL

What should we do?

Theorem (J - Montagna 2008)

The varieties ICGBL and CGBL have the finite model property (FEP), hence their universal theory is decidable

- Currently there is no "reasonable" proof procedure
- Can use an automated theorem prover and finite model finder
- E.g. Prover9 and Mace4
- Same works (in principle) whenever we have FEP or FMP but not really effective

Deciding equations in GBL?

GBL is short for *generalized basic logic algebras*

Suppose someone has a GBL-term t and asks if $t = 1$ is true in GBL

What should we do?

- Currently don't know if GBL has a decidable equational theory
- Can still use an automated theorem prover and finite model finder but there are equations that have no finite counter model
- E.g. every finite GBL-algebra is commutative [J - Montagna 2006]
- More effective to send t to Nick Galatos

Deciding equations in GBL?

GBL is short for *generalized basic logic algebras*

Suppose someone has a GBL-term t and asks if $t = 1$ is true in GBL

What should we do?

- Currently don't know if GBL has a decidable equational theory
- Can still use an automated theorem prover and finite model finder but there are equations that have no finite counter model
- E.g. every finite GBL-algebra is commutative [J - Montagna 2006]
- More effective to send t to Nick Galatos

Deciding equations in GBL?

GBL is short for *generalized basic logic algebras*

Suppose someone has a GBL-term t and asks if $t = 1$ is true in GBL

What should we do?

- Currently don't know if GBL has a decidable equational theory
- Can still use an automated theorem prover and finite model finder but there are equations that have no finite counter model
- E.g. every finite GBL-algebra is commutative [J - Montagna 2006]
- More effective to send t to Nick Galatos

Deciding equations in GBL?

GBL is short for *generalized basic logic algebras*

Suppose someone has a GBL-term t and asks if $t = 1$ is true in GBL

What should we do?

- Currently don't know if GBL has a decidable equational theory
- Can still use an automated theorem prover and finite model finder but there are equations that have no finite counter model
- E.g. every finite GBL-algebra is commutative [J - Montagna 2006]
- More effective to send t to Nick Galatos

Deciding equations in GBL?

GBL is short for *generalized basic logic algebras*

Suppose someone has a GBL-term t and asks if $t = 1$ is true in GBL

What should we do?

- Currently don't know if GBL has a decidable equational theory
- Can still use an automated theorem prover and finite model finder but there are equations that have no finite counter model
- E.g. every finite GBL-algebra is commutative [J - Montagna 2006]
- More effective to send t to Nick Galatos

Questions

What algorithm should be used to decide quasi-equations in CGBL?

Theorem (J - Montagna 2008)

The variety of GBL-algebras has undecidable quasiequational theory

Is the equational theory of GBL decidable?

Is there a cut-free sequent (or hypersequent) system for CGBL (or GBL)?

Questions

What algorithm should be used to decide quasi-equations in CGBL?

Theorem (J - Montagna 2008)

The variety of GBL-algebras has undecidable quasiequational theory

Is the equational theory of GBL decidable?

Is there a cut-free sequent (or hypersequent) system for CGBL (or GBL)?

Questions

What algorithm should be used to decide quasi-equations in CGBL?

Theorem (J - Montagna 2008)

The variety of GBL-algebras has undecidable quasiequational theory

Is the equational theory of GBL decidable?

Is there a cut-free sequent (or hypersequent) system for CGBL (or GBL)?

Questions

What algorithm should be used to decide quasi-equations in CGBL?

Theorem (J - Montagna 2008)

The variety of GBL-algebras has undecidable quasiequational theory

Is the equational theory of GBL decidable?

Is there a cut-free sequent (or hypersequent) system for CGBL (or GBL)?

Questions

What algorithm should be used to decide quasi-equations in CGBL?

Theorem (J - Montagna 2008)

The variety of GBL-algebras has undecidable quasiequational theory

Is the equational theory of GBL decidable?

Is there a cut-free sequent (or hypersequent) system for CGBL (or GBL)?

Questions

What algorithm should be used to decide quasi-equations in CGBL?

Theorem (J - Montagna 2008)

The variety of GBL-algebras has undecidable quasiequational theory

Is the equational theory of GBL decidable?

Is there a cut-free sequent (or hypersequent) system for CGBL (or GBL)?

References

[N. Galatos, P. Jipsen, T. Kowalski, H. Ono, 2007] "Residuated Lattices: An algebraic glimpse at substrutural logics", Studies in Logic, vol 151, Elsevier, 2007, xxi+509 pp

[P. Jipsen, F. Montagna, 2006] *On the structure of generalized BL-algebras*, Algebra Universalis **55** (2006), 226–237

[P. Jipsen, F. Montagna, 2008] *The Blok-Ferreirim theorem for normal GBL-algebras and its application*, Algebra Universalis, to appear

[P. Jipsen, C. Tsinakis, 2002] *A survey on residuated lattices*, in: J. Martinez (Ed.), Ordered Algebraic Structures, Kluwer, 2002, 19–56