Algebraic Proof Mining for Fast Decision Procedures

Aaron Stump

Computer Science and Engineering
Washington University
St. Louis, Missouri, USA

CSE Retreat ’05

CAREER: Semantic Programming

Develop language-based approaches to program verification.

Programmer knows why code is right.

Provide language constructs to encode that knowledge formally.

"Programming with proofs":

- Code intertwined with proofs.
- Functions input proofs of pre-conditions.
- They output proofs of post-conditions.
- Proofs built by hand, with automated help.

Goal: make provably correct coding a practical reality!

Case Study: Decision Procedures

Decision procedures (DPs) check validity of logical formulas.

Can handle background theories: arithmetic, arrays, bitvectors.

Used for algorithmic verification.

Good case study for programming with proofs:

- Relations between proofs and DPs well understood.
- Proofs independently valuable.
- Proofs are about data (formulas), not executions.
- Proofs can be enormous, so good engineering required.

Leverages PI expertise in the area.

Decision Procedures: Hot Topic

Decision procedures are hot in verification.

Increasing submissions at CAV and TACAS.

Barrett, de Moura, and Stump organize SMT-COMP 2005:

- “Satisfiability Modulo Theories” Competition.
- Satellite event of CAV 2005, Edinburgh, Scotland.
- 12 solvers from Europe and U.S.
- Spurred collection of 1338 benchmarks, in 7 theories.
- Hot tools: Barcelogic Tools (T.U. Catalonia), Yices (SRI).
- Industrial interest from Intel Strategic CAD Labs, NEC Labs.

Inside a Decision Procedure

SAT solver: ANDs, ORs, and NOTs.

Theory solver: $3x + y < z$, $write(a, i, v) = b$, or just $a = b$.

SAT solver satisfies boolean structure, then calls theory solver.

SAT solver

Theory Reasoner

Inside a Decision Procedure

SAT solver: ANDs, ORs, and NOTs.

Theory solver: $3x + y < z$, $write(a, i, v) = b$, or just $a = b$.

SAT solver satisfies boolean structure, then calls theory solver.

$$\text{SAT solver} \xrightarrow{\begin{array}{l} a = b \\ d = b \\ b = c \\ a \neq c \end{array}} \text{Theory Reasoner}$$

Inside a Decision Procedure

SAT solver: ANDs, ORs, and NOTs.

Theory solver: $3x + y < z$, $write(a, i, v) = b$, or just $a = b$.

SAT solver satisfies boolean structure, then calls theory solver.

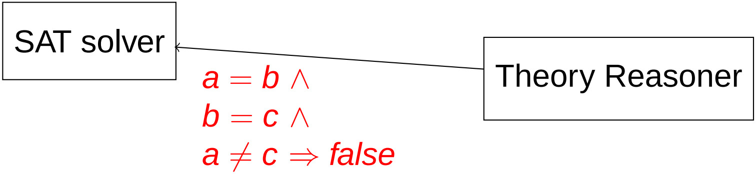
$$\boxed{\text{SAT solver}} \xrightarrow{\substack{a = b \\ d = b \\ b = c \\ a \neq c}} \boxed{\text{Theory Reasoner}}$$

Inside a Decision Procedure

SAT solver: ANDs, ORs, and NOTs.

Theory solver: $3x + y < z$, $write(a, i, v) = b$, or just $a = b$.

SAT solver satisfies boolean structure, then calls theory solver.



Inside a Decision Procedure

SAT solver: ANDs, ORs, and NOTs.

Theory solver: $3x + y < z$, $write(a, i, v) = b$, or just $a = b$.

SAT solver satisfies boolean structure, then calls theory solver.

SAT solver

Theory Reasoner

Conflict clause:
$a \neq b \ \vee \ b \neq c \ \vee \ a = c$

Conflict Clauses

Conflict clauses prune later search.

Smaller clauses aren't always better [Malik et al. 2001].

A subset of a clause **is** always better.

Recent work: subsets of conflict clauses for EUF solvers [Nieuwenhuis and Oliveras 2005, Stump and Tan 2005, de Moura et al. 2004].

Proof Mining [Barrett,Dill,Stump 2002]

Suppose theory solver produces proof of contradiction.

Just return the assumptions used in that proof.

Assumptions: $a = b,\ d = b,\ b = c,\ a \neq c$.

Proof:

$$\dfrac{\dfrac{a = b \qquad b = c}{a = c}\,\text{Trans} \qquad a \neq c}{\mathit{false}}\,\text{Contra}$$

Proof Mining [Barrett,Dill,Stump 2002]

Suppose theory solver produces proof of contradiction.

Just return the assumptions used in that proof.

Assumptions: $a = b,\ d = b,\ b = c,\ a \neq c$.

Proof:

$$\dfrac{\dfrac{a = b \qquad b = c}{a = c}\,\text{Trans} \qquad a \neq c}{\mathit{false}}\,\text{Contra}$$

Proof Mining [Barrett,Dill,Stump 2002]

Suppose theory solver produces proof of contradiction.

Just return the assumptions used in that proof.

Assumptions: $a = b,\ d = b,\ b = c,\ a \neq c$.

Proof:

$$\dfrac{\dfrac{a = b \qquad b = c}{a = c}\,\text{Trans} \qquad a \neq c}{\mathit{false}}\,\text{Contra}$$

Flabby Proofs from Union-Find

Assumptions: $a = b,\ a = c,\ a \neq c$.

Union-find structure: Action:

Flabby Proofs from Union-Find

Assumptions: $a = b,\ a = c,\ a \neq c$.

Union-find structure: Action: union(a,b)

$$\begin{array}{cc} & b \\ \nearrow & \\ a & \end{array}$$

Flabby Proofs from Union-Find

Assumptions: $a = b,\ a = c,\ a \neq c$.

Union-find structure: Action: union(a,c)

$$\begin{array}{ccc} & b & \\ \nearrow & & \nwarrow \\ a & & c \end{array}$$

Flabby Proofs from Union-Find

Assumptions: $a = b,\ a = c,\ a \neq c$.

Union-find structure: find(a) = b, find(c) = b

$$\begin{tikzcd} & b & \\ a \arrow[ur] & & c \arrow[ul] \end{tikzcd}$$

Flabby Proofs from Union-Find

Assumptions: $a = b,\ a = c,\ a \neq c$.

Union-find structure: find(a) = b, find(c) = b

$$\begin{array}{ccc} & b & \\ \nearrow & & \nwarrow \\ a & & c \end{array}$$

Resulting flabby proof:

$$\dfrac{\dfrac{a = b \qquad \dfrac{\dfrac{a = b}{b = a}\text{Symm} \qquad a = c}{b = c}\text{Trans}}{a = c}\text{Trans} \qquad a \neq c}{\mathit{false}}\text{Contra}$$

Flabby Proofs from Union-Find

Assumptions: $a = b,\ a = c,\ a \neq c$.

Union-find structure: find(a) = b, find(c) = b

$$\begin{array}{ccc} & b & \\ \nearrow & & \nwarrow \\ a & & c \end{array}$$

Resulting flabby proof:

$$\dfrac{\dfrac{a = b \quad \dfrac{\dfrac{a = b}{b = a}\text{Symm} \quad a = c}{b = c}\text{Trans}}{a = c}\text{Trans} \quad a \neq c}{\mathit{false}}\text{Contra}$$

Algebraic Proof Mining [Stump and Tan 2005]

Idea: transform proofs to get rid of flab.

Algebraic: transformations based on equations between proofs.

$$\dfrac{\dfrac{\stackrel{\mathcal{D}_1}{a = b} \quad \stackrel{\mathcal{D}_2}{b = c}}{a = c}\text{Trans} \qquad \stackrel{\mathcal{D}_3}{c = d}}{a = d} \quad \cong \quad \dfrac{\stackrel{\mathcal{D}_1}{a = b} \qquad \dfrac{\stackrel{\mathcal{D}_2}{b = c} \quad \stackrel{\mathcal{D}_3}{c = d}}{b = d}\text{Trans}}{a = d}$$

$$\dfrac{\dfrac{}{a = a}\text{Refl} \qquad \stackrel{\mathcal{D}}{a = b}}{a = b}\text{Trans} \quad \cong \quad \stackrel{\mathcal{D}}{a = b}$$

$$\dfrac{\dfrac{\stackrel{\mathcal{D}}{a = b}}{b = a}\text{Symm} \qquad \stackrel{\mathcal{D}}{a = b}}{b = b}\text{Trans} \quad \cong \quad \dfrac{}{b = b}\text{Refl}$$

An Equational Theory On Proof Terms

Equations

Trans(Trans(d1,d2),d3)	$\cong$	Trans(d1,Trans(d2,d3))
Trans(Refl,d)	$\cong$	d
Trans(Symm(d),d)	$\cong$	Refl

Which equational theory is it?

This Is Equational Group Theory

Compare these equations:

Trans(Trans(d1,d2),d3)	$\cong$	Trans(d1,Trans(d2,d3))
Trans(Refl,d)	$\cong$	d
Trans(Symm(d),d)	$\cong$	Refl

With the group axioms:

$$
\begin{aligned}
(d_1 * d_2) * d_3 &\cong d_1 * (d_2 * d_3) \\
1 * d &\cong d \\
d^{-1} * d &\cong 1
\end{aligned}
$$

Trans is $*$, Symm is $()^{-1}$, and Refl is 1.

Transforming Equality Proofs

Theorem (Knuth-Bendix, 1970)

These rules put every group term into canonical form:

$$
\begin{array}{lll}
1. & (x * y) * z & \rightarrow & x * (y * z) \\
2. & x^{-1} * x & \rightarrow & 1 \\
3. & x * x^{-1} & \rightarrow & 1 \\
4. & x * (x^{-1} * y) & \rightarrow & y \\
5. & x^{-1} * (x * y) & \rightarrow & y \\
6. & (x * y)^{-1} & \rightarrow & y^{-1} * x^{-1} \\
7. & 1 * x & \rightarrow & x \\
8. & x * 1 & \rightarrow & x \\
9. & 1^{-1} & \rightarrow & 1 \\
10. & (x^{-1})^{-1} & \rightarrow & x
\end{array}
$$

Rewrite proofs to remove flab.

Better: mine assumptions without actually rewriting.

Empirical Results in CVC

Benchmark	dec. orig	time orig (s)	dec. mining	time mining (s)
dlx-regfile	2807	2.1	2430	2.5
dlx-dmem	1336	1.0	1048	0.9
pp-regfile	115197	295.7	44547	121.9
pp-dmem	25928	68.1	11899	23.7
pp-bloaddata	4060	1.7	3461	2.1
pp-TakenBranch	15364	26.1	11928	24.6

Conclusion

Decision procedures: case study for programming with proofs.

Algebraic proof mining: mine info from transformed proofs.

For equality proofs, use rewrite rules for free group theory.

2x performance improvement on large benchmarks observed.

Future work: apply to other theories.

Future work: RVC decision procedure, written in RSP.

Congruence Rules

Congruence rules are commuting endomorphisms.

These proofs prove the same theorem:

$$\dfrac{\dfrac{a = b \qquad b = c}{a = c}\,\text{Trans}}{f(a,d) = f(c,d)}\,\text{Cong}_{f,1}$$

$$\dfrac{\dfrac{a = b}{f(a,d) = f(b,d)}\,\text{Cong}_{f,1} \qquad \dfrac{b = c}{f(b,d) = f(c,d)}\,\text{Cong}_{f,1}}{f(a,d) = f(c,d)}\,\text{Trans}$$

Commutativity is also required:

$$\text{Trans}(\text{Cong}_{f,1}(\text{d1}),\text{Cong}_{f,2}(\text{d2})) \cong \text{Trans}(\text{Cong}_{f,2}(\text{d2}),\text{Cong}_{f,1}(\text{d1}))$$

Rules for Commuting Endomorphisms

1.	$(x * y) * z$	$\longrightarrow$	$x * (y * z)$	11.	$f(1)$	$\longrightarrow$	1
2.	$x^{-1} * x$	$\longrightarrow$	1	12.	$(f(x))^{-1}$	$\longrightarrow$	$f(x^{-1})$
3.	$x * x^{-1}$	$\longrightarrow$	1	13.	$f(x) * f(y)$	$\longrightarrow$	$f(x * y)$
4.	$x * (x^{-1} * y)$	$\longrightarrow$	y	14.	$f(x) * (f(y) * z)$	$\longrightarrow$	$f(x * y) * z$
5.	$x^{-1} * (x * y)$	$\longrightarrow$	y	15.	$g(1)$	$\longrightarrow$	1
6.	$(x * y)^{-1}$	$\longrightarrow$	$y^{-1} * x^{-1}$	16.	$(g(x))^{-1}$	$\longrightarrow$	$g(x^{-1})$
7.	$1 * x$	$\longrightarrow$	x	17.	$g(x) * g(y)$	$\longrightarrow$	$g(x * y)$
8.	$x * 1$	$\longrightarrow$	x	18.	$g(x) * (g(y) * z)$	$\longrightarrow$	$g(x * y) * z$
9.	1^{-1}	$\longrightarrow$	1	19.	$f(x) * g(y)$	$\longrightarrow$	$g(y) * f(x)$
10.	$(x^{-1})^{-1}$	$\longrightarrow$	x	20.	$f(x) * (g(y) * z)$	$\longrightarrow$	$g(y) * (f(x) * z)$