Abstractions via Compression

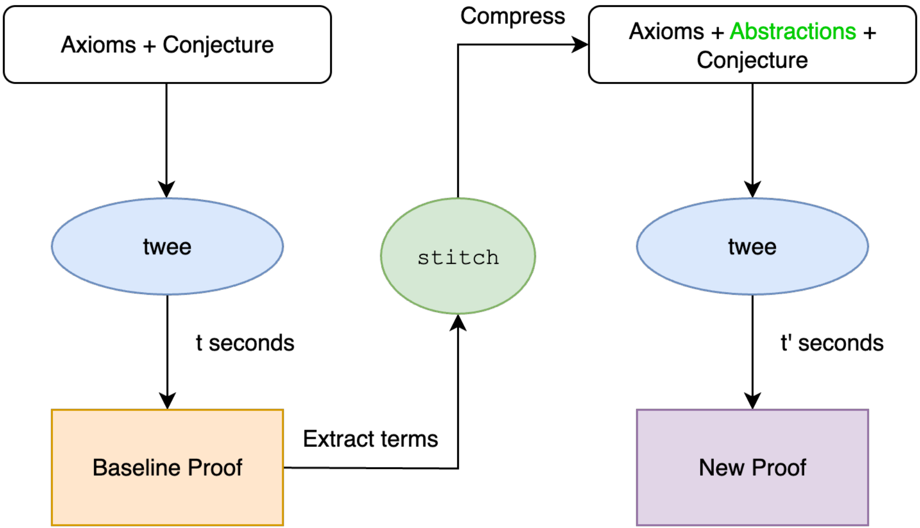
Speeding up equational solving with definitional extensions

Guy Axelrod, Nick Smallbone, Moa Johansson

Chalmers University of Technology

September 2025

Abstractions via Compression (Overview)



We define good abstractions as those for which $t' << t$

Setting: equational solvers

Input problem $P = (\mathcal{E}, \mathcal{C})$. Axiom(s) ($\mathcal{E}$) and conjecture(s) ($\mathcal{C}$).

Example input problem:

Axioms:

$$
\begin{aligned}
&X + (Y + Z) = (X + Y) + Z \\
&0 + X = X \\
&X + 0 = X \\
&(-X) + X = 0 \\
&X + (-X) = 0 \\
&a + b = a
\end{aligned}
$$

Conjecture:

$$b = 0$$

Setting: equational solvers

Output (if terminates): proof $\Gamma(P)$.

Example proof:

```
Axiom 1 (assumption): a + b = a.
Axiom 2 (plus_zero): 0 + X = X.
Axiom 3 (minus_left): -X + X = 0.
Axiom 4 (associativity): X + (Y + Z) = (X + Y) + Z.

Conjecture: b = 0.
Proof:
  b
= { by axiom 2 (plus_zero) R->L }
  0 + b
= { by axiom 3 (minus_left) R->L }
  (-a + a) + b
= { by axiom 4 (associativity) R->L }
  -a + (a + b)
= { by axiom 1 (assumption) }
  -a + a
= { by axiom 3 (minus_left) }
  0
```

Abstraction

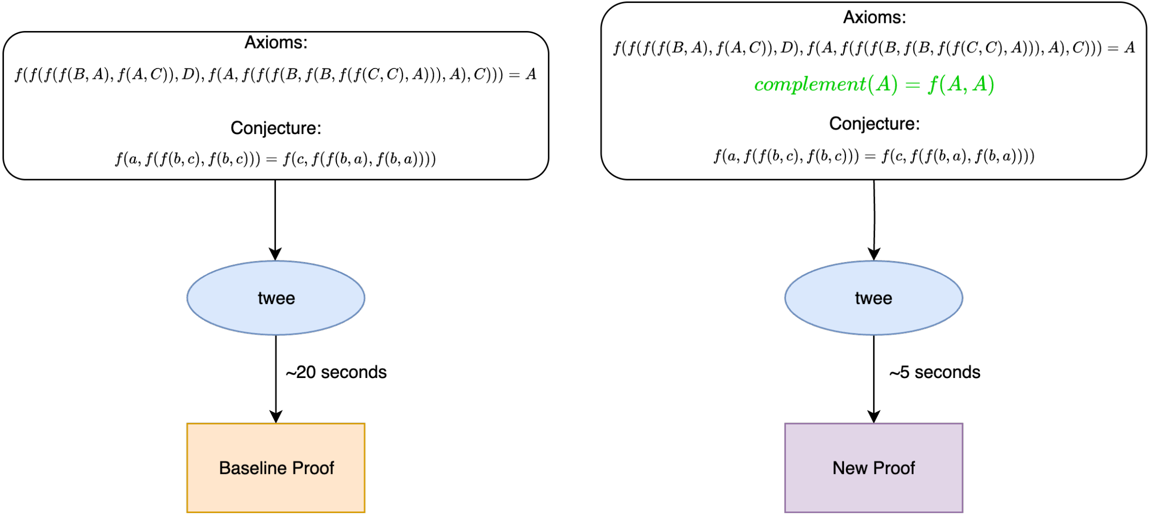
- Input problem $P = (\mathcal{E}, \mathcal{C})$. Axiom(s) $(\mathcal{E})$ and conjecture(s) $(\mathcal{C})$.
- Output (if terminates): proof $\Gamma(P)$.
- **Abstraction**: A definitional extension.

Definitional extension

Introduce fresh function symbol g.
An ***abstraction*** is an identity of the form $g(\bar{x}) \approx t(\bar{x})$, which we add to $\mathcal{E}$ to get the augmented problem $P' = (\mathcal{E} \cup \{g(\bar{x}) \approx t(\bar{x})\}, \mathcal{C})$.
This preserves the original theory but may speed up search.

A ***good abstraction*** is one that leads to $\Gamma(P')$ being found quicker than $\Gamma(P)$.

Example:


Axioms:
$f(f(f(f(B,A),f(A,C)),D),f(A,f(f(f(B,f(B,f(f(C,C),A))),A),C))) = A$
Conjecture:
$f(a,f(f(b,c),f(b,c))) = f(c,f(f(b,a),f(b,a)))$
twee
~20 seconds
Baseline Proof
Axioms:
$f(f(f(f(B,A),f(A,C)),D),f(A,f(f(f(B,f(B,f(f(C,C),A))),A),C))) = A$
$complement(A) = f(A,A)$
Conjecture:
$f(a,f(f(b,c),f(b,c))) = f(c,f(f(b,a),f(b,a)))$
twee
~5 seconds
New Proof

`twee` and Conjecture Flattening

Conjecture Flattening

For every function term $f(\ldots)$ appearing in the conjecture, introduce a fresh constant symbol a and add the constant abstraction $a \approx f(\ldots)$ [a] to the axioms.

[a] Internally, `twee` works with ground conjectures.

Example:
if the (ground) conjecture is $f(g(a), b) \approx h(c)$, Conjecture flattening adds the abstractions:

- $d_1 \approx f(g(a), b)$
- $d_2 \approx g(a)$
- $d_3 \approx h(c)$

Aims

- **Ultimate goal:** given only P, generate good abstractions.
- **Weaker goal (this talk):** given P **and** a baseline proof $\Gamma(P)$, generate good abstractions.

Weaker goal can be seen as part of the broader problem of proof improvement and refactoring.

Related Work

Related proof improvement and refactoring work;

- https://github.com/JUrban/E_conj : Finding useful "cuts" in E proofs.
- Veroff, R.: Solving open questions and other challenge problems using proof sketches.
- Puzis, Y., Gao, Y., Sutcliffe, G.: Automated generation of interesting theorems.
- Vyskočil, J., Štěpánek, P.: Improving efficiency of prolog programs by fully automated unfold/fold transformation.
- Vyskočil, J., Stanovský, D., Urban J.: Automated Proof Compression by Invention of New Definitions.

Compression of Proofs

Compression

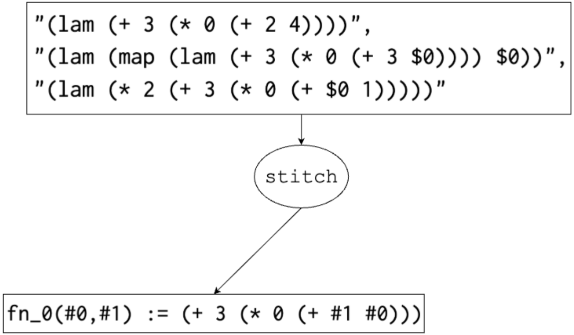
Given some set of terms T, ***compression*** refers to finding a definition A that minimizes the objective:

$$size(A) + \sum_{t \in T} size(rewrite(t, A))$$

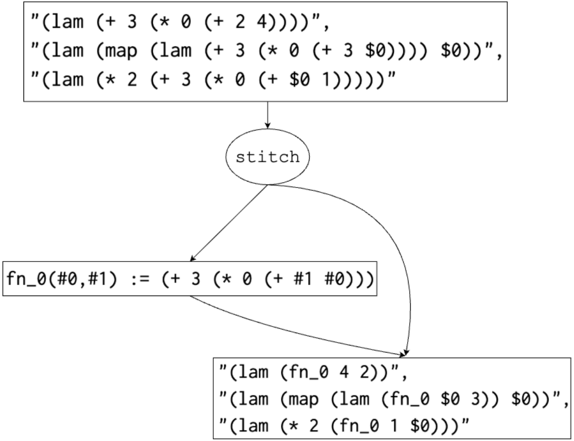
Given a proof $\Gamma(P)$,

- Extract some set $T_{\Gamma(P)}$ of proof terms
- Generate abstraction A via compression.

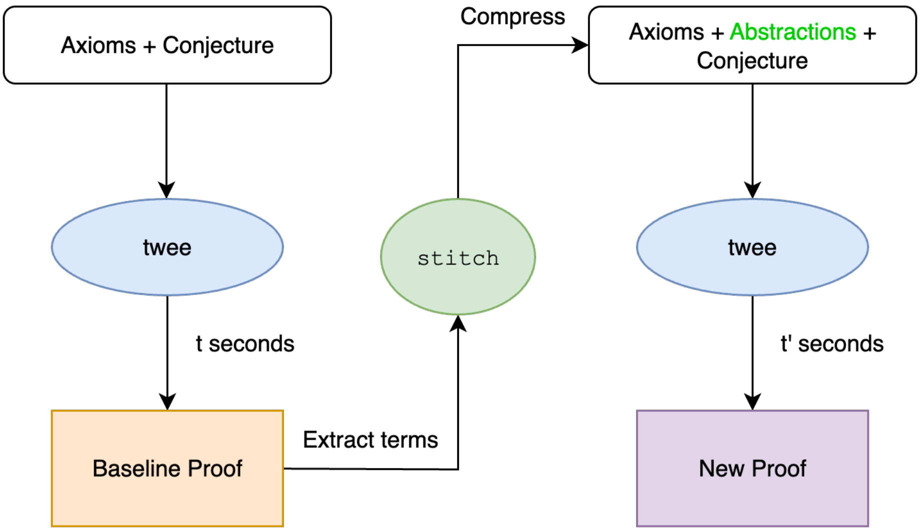
stitch



stitch


"(lam (+ 3 (* 0 (+ 2 4))))",
"(lam (map (lam (+ 3 (* 0 (+ 3 $0)))) $0))",
"(lam (* 2 (+ 3 (* 0 (+ $0 1)))))"
stitch
fn_0(#0,#1) := (+ 3 (* 0 (+ #1 #0)))
"(lam (fn_0 4 2))",
"(lam (map (lam (fn_0 $0 3)) $0))",
"(lam (* 2 (fn_0 1 $0)))"

Experiments

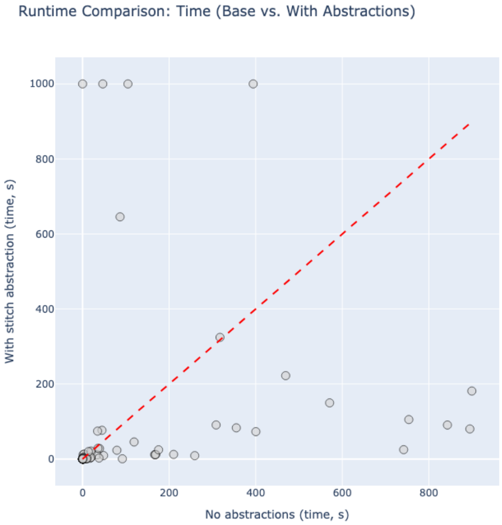


TPTP Unit EQuality problems (UEQ):

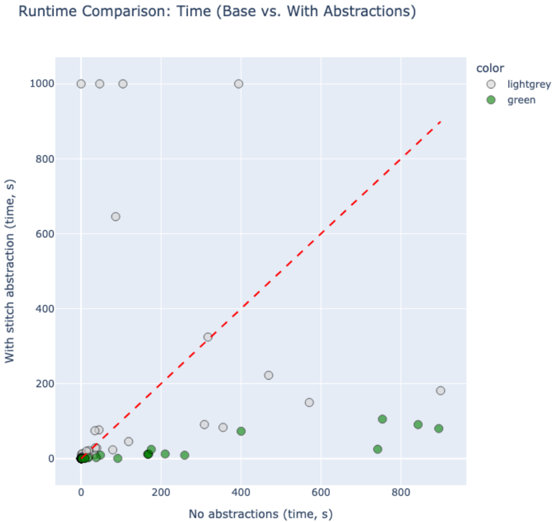
Theory	Count
ALG (General Algebra)	18
BOO (Boolean Algebra)	53
COL (Combinatory Logic)	117
GRP (Group Theory)	428
LAT (Lattice Theory)	110
LCL (Logic Calculi)	58
REL (Relation Algebra)	79
RNG (Ring Theory)	46
ROB (Robbins Algebra)	23
Total	932

Table: Number of problems per theory

LAT


Runtime Comparison: Time (Base vs. With Abstractions)
With stitch abstraction (time, s)
No abstractions (time, s)
0
200
400
600
800
1000
0
200
400
600
800

LAT (5x speedup green)



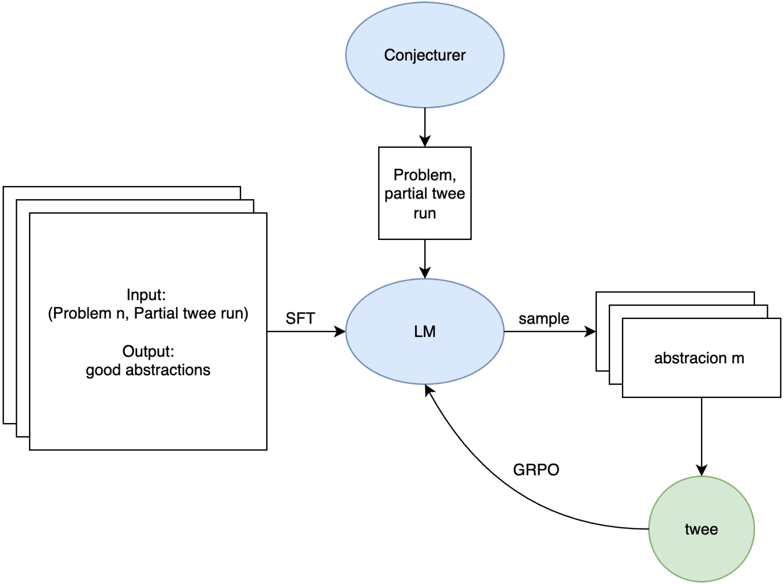
Results

- 59,648 total runs. Corresponding to 932 problems, each run with 64 different hyperparameter configurations.
- 859 of the 932 problems were solved by `twee` with no abstractions (given 1000 second timeout).
- Of these 859 problems, we found that our approach led us to finding abstractions that yielded:
  - 2x speedup in $\sim$60% of problems.
  - 5x speedup in $\sim$30% of problems.
  - 10x speedup in $\sim$20% of problems.
  - 20x speedup in $\sim$10% of problems.

Future Work

- Experiment with compression objective, weighting proof terms based on some measure of relevance.
- Common abstractions per theory, applied to hard problems.
- Explore Time-slicing: plain Twee + abstractions-Twee.
- Anti-unification over E-graphs instead of compression.
- Use as means to produce training data. Bootstrap RL loop towards generative model capable of tackling the ultimate goal (i.e. abstraction proposer that works from P without $\Gamma(P)$).

Train Model


Conjecturer
Problem, partial twee run
Input:
(Problem n, Partial twee run)
Output:
good abstractions
SFT
LM
sample
abstracion m
GRPO
twee

Future Work + Questions

- Experiment with compression objective, weighting proof terms based on some measure of relevance.
- Common abstractions per theory, applied to hard problems.
- Explore Time-slicing: plain Twee + abstractions-Twee.
- Anti-unification over E-graphs instead of compression.
- Use as means to produce training data. Bootstrap RL loop towards generative model capable of tackling the ultimate goal (i.e. abstraction proposer that works from P without $\Gamma(P)$).

Thank you!
Questions?