Automating it

simp and auto

simp rewriting and a bit of arithmetic

auto rewriting and a bit of arithmetic, logic & sets

simp and auto

simp rewriting and a bit of arithmetic

auto rewriting and a bit of arithmetic, logic & sets

- Show you where they got stuck

simp and auto

simp rewriting and a bit of arithmetic

auto rewriting and a bit of arithmetic, logic & sets

- Show you where they got stuck
- highly incomplete wrt logic

blast

- A complete (for FOL) tableaux calculus implementation

blast

- A complete (for FOL) tableaux calculus implementation
- Covers logic, sets, relations, . . .

blast

- A complete (for FOL) tableaux calculus implementation
- Covers logic, sets, relations, . . .
- Extensible with intro/elim rules

blast

- A complete (for FOL) tableaux calculus implementation
- Covers logic, sets, relations, . . .
- Extensible with intro/elim rules
- Almost no “=”

Demo: blast

blast: A 2-stage implementation

1. Search for proof with quick, dirty and possibly buggy program

blast: A 2-stage implementation

1. Search for proof with quick, dirty and possibly buggy program (while recording proof tree)

blast: A 2-stage implementation

1. Search for proof with quick, dirty and possibly buggy program (while recording proof tree)
2. Check proof tree with Isabelle kernel

blast: A 2-stage implementation

1. Search for proof with quick, dirty and possibly buggy program (while recording proof tree)
2. Check proof tree with Isabelle kernel

This is the *LCF-architecture* by Robin Milner:

blast: A 2-stage implementation

1. Search for proof with quick, dirty and possibly buggy program (while recording proof tree)
2. Check proof tree with Isabelle kernel

This is the *LCF-architecture* by Robin Milner:

- Proofs only constructable via small trustworthy kernel

blast: A 2-stage implementation

1. Search for proof with quick, dirty and possibly buggy program (while recording proof tree)
2. Check proof tree with Isabelle kernel

This is the *LCF-architecture* by Robin Milner:

- Proofs only constructable via small trustworthy kernel
- Proof search programmable in ML on top

blast: A 2-stage implementation

1. Search for proof with quick, dirty and possibly buggy program (while recording proof tree)
2. Check proof tree with Isabelle kernel

This is the *LCF-architecture* by Robin Milner:

- Proofs only constructable via small trustworthy kernel
- Proof search programmable in ML on top

Isabelle follows the LCF architecture

fast and friends

fast slow and incomplete version of *blast*

fast and friends

fast slow and incomplete version of *blast*

fastsimp rewriting and logic

fast and friends

fast slow and incomplete version of *blast*

fastsimp rewriting and logic

force slower but completer version of *fastsimp*

metis

- An fast resolution theorem prover in ML

metis

- An fast resolution theorem prover in ML
- Can deal with bidirectional “=”

metis

- An fast resolution theorem prover in ML
- Can deal with bidirectional “=”
- Knows only pure logic, not sets etc

Demo: beyond blast

Problems

- Most proofs require additional lemmas.

Problems

- Most proofs require additional lemmas.
- Adding arbitrary lemmas slows *blast* down significantly;

Problems

- Most proofs require additional lemmas.
- Adding arbitrary lemmas slows *blast* down significantly; *metis* copes better.

Problems

- Most proofs require additional lemmas.
- Adding arbitrary lemmas slows *blast* down significantly; *metis* copes better.
- Finding the right lemmas in a library of thousands of lemmas is light years beyond *blast* and *metis*.

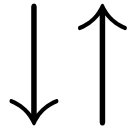
Problems

- Most proofs require additional lemmas.
- Adding arbitrary lemmas slows *blast* down significantly; *metis* copes better.
- Finding the right lemmas in a library of thousands of lemmas is light years beyond *blast* and *metis*.
- There are highly optimized *ATPs* (automatic theorem provers) for FOL that can deal with large libraries . . .

Sledgehammer

Architecture

Isabelle

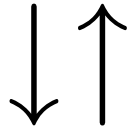


ATPs

Architecture

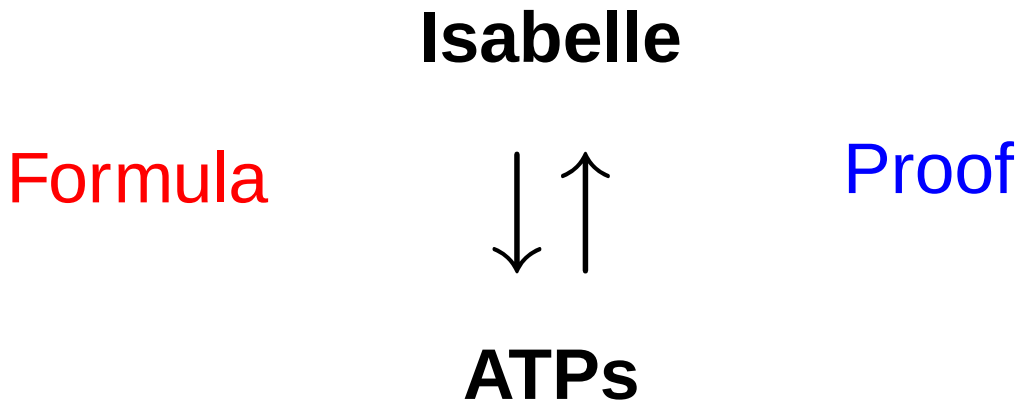
Isabelle

Formula

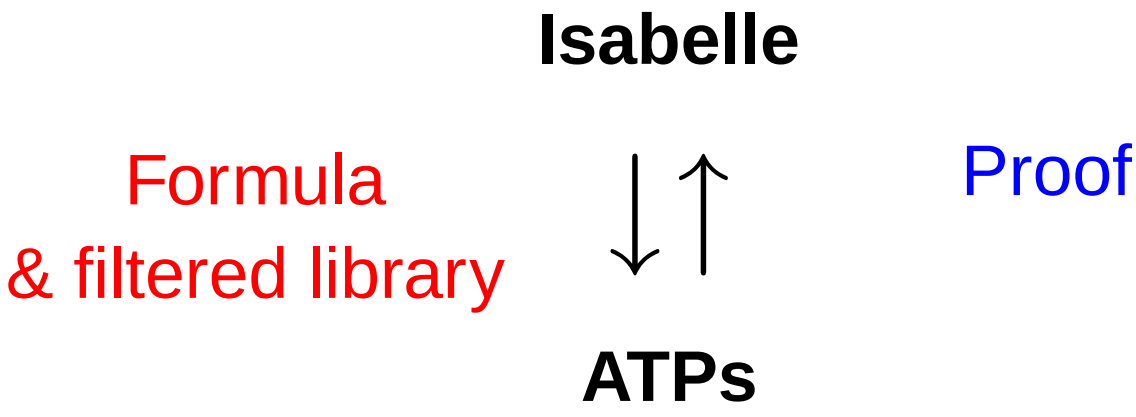


ATPs

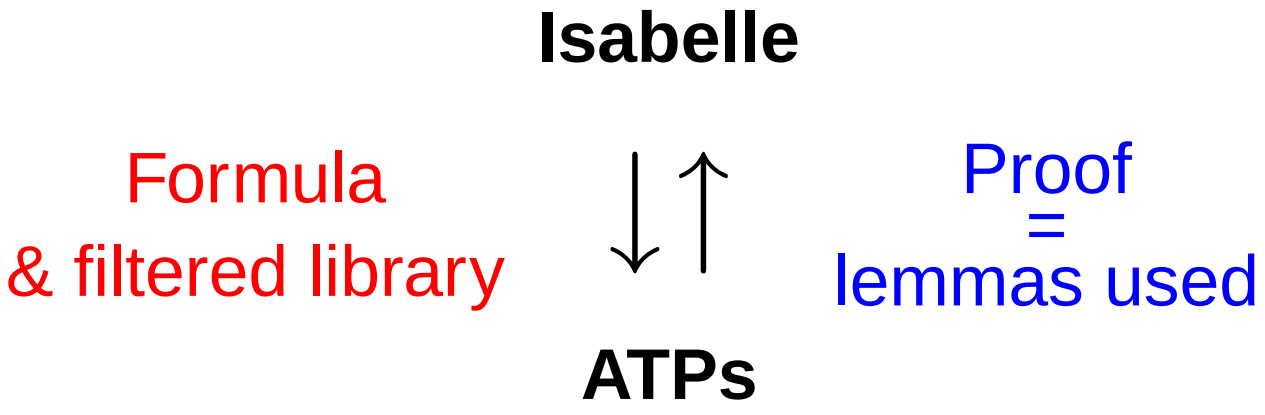
Architecture



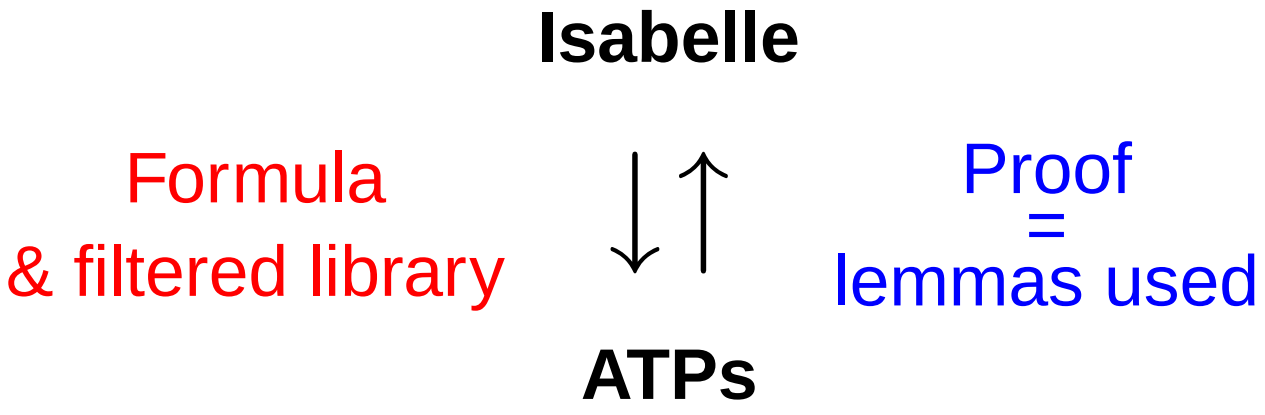
Architecture



Architecture



Architecture



Empirical study:

Sledgehammer works for 1/3 of non-trivial Isabelle proofs

Demo: Sledghehammer

Automating Arithmetic

Automating arithmetic

arith:

Automating arithmetic

arith:

- proves linear formulas (no “$*$”)

Automating arithmetic

arith:

- proves linear formulas (no “$*$”)
- complete for quantifier-free *real* arithmetic

Automating arithmetic

arith:

- proves linear formulas (no "$*$")
- complete for quantifier-free *real* arithmetic
- complete for first-order theory of *nat* and *int* (Presburger arithmetic)

Automating arithmetic

Theorem (Tarski) $Th(\mathbb{R}, +, -, *, <, =)$ is decidable.

Automating arithmetic

Theorem (Tarski) $Th(\mathbb{R}, +, -, *, <, =)$ is decidable.

An incomplete but (often) fast method for the quantifier-free fragement:

sos

Automating arithmetic

Theorem (Tarski) $Th(\mathbb{R}, +, -, *, <, =)$ is decidable.

An incomplete but (often) fast method for the quantifier-free fragement:

sos

Idea: (re)write polynomials as sums-of-squares to prove non-negativity

Demo: Arithmetic