# Revisiting SAD

Steffen Frerix    Peter Koepke

University of Bonn

AITP, March 2018

# Two goals of formalizing mathematics

1. Model a (local) reasoning process

2. Build a body of knowledge grounded in axioms

# SAD - rework of core features

- Evidence collection
- Definition unfolding
- Thesis management

# Expressiveness of ForTheL - Sets and Functions

```
Signature.  A set is a notion.

Signature.  Let M be a set.  An element of M is a
     notion.

Signature.  A function is a notion.

Signature.  Let f be a function.  Dom(f) is a set.

Signature.  Let f be a function.  Let x be an element
     of Dom(f).  f[x] is an object.
```

Problem: schemes cannot be expressed

# Expressiveness of ForTheL - Proof operations

Four kinds of operations in a proof

- Assumption
- Affirmation
- Choice
- Case hypothesis

New operation: Definition

```
proof.  Let m be a natural number.
Define M = { n in NAT | n is a divisor of m }.
Let g be a function from NAT to NAT.
Define f[n] = n + g[n] for n in M.
⋮
```

# Expressiveness of ForTheL - Computations

enable syntax for expressing computational reasoning

- equation chains
- supported by a simple rewrite tool

automatic extraction of rewrite rules

- rules are simplified with local properties
- extension of modularity to rewriting

# ForTheL - Types and Notions

ForTheL notions form soft types.

```
Axiom.  Every nonzero real has an inverse.
```

The translation to FOL is done using type guards.

$$\forall v_0 \, \mathsf{aReal}(v_0) \wedge \mathsf{isNonzero}(v_0) \rightarrow \exists v_1 \, \mathsf{aInverseOf}(v_1, v_0)$$

The result: clutter in the input of the ATP

# ForTheL - Translation to many-sorted logic

Approach: Translate ForTheL into (polymorphic) many-sorted logic instead of FOL.

- provers have inbuild support for (at least monomorphic) many-sorted logic
- there exist well-performing encodings to FOL for (polymorphic) many-sorted logic

This has several drawbacks

- the user must decide which notions become sorts of the logic
- casting functions are necessary to reflect the sort hierarchy
- inflexible: a notion may be used as a sort in one context and as a proper predicate in another

# ForTheL - Recognizing typings

Instead: Use the ontological information in of a ForTheL text to decide where a predicate in its FOL image plays the role of a typing and can be soundly deleted

- realized in a calculus
- decision on a per formula basis: reductions cannot become unsound through extension of the text
- not limited to unary predicates (or even predicates)

# Example: Chinese Remainder Theorem

```
Theorem.  Let I,J be ideals.  Suppose that every element
belongs to I + J. For all elements x,y there exists an
element w such that w = x (mod I) and w = y (mod J).
```

```
Theorem.  Let a,b be elements.  Assume that a is nonzero or
b is nonzero.  Let c be a gcd of a and b.  Then c is an
element of <a> + <b>.
```

| **Version** | **goals** | **prover total** | **prover max** |
|---|---|---|---|
| SAD 2.3 (SPASS 3.5) | 51 | 2:05.29 | 0:00.36 |
| SAD dev (Eprover 2.0) | 51 | 0:00.75 | 0:00.05 |

# Example: Sequences in an ordered field

- A background text about ordered fields (OF)
- A short text introducing a notion of natural number and the archimedean axiom (Nat)
- A text about sequences in an ordered field (Seq)

| **Text** | **total statements** | **derived** | **axioms/defs** |
|---|---|---|---|
| OF | 77 | 53 | 24 |
| Nat | 11 | 3 | 8 |
| Seq | 16 | 6 | 10 |

# Example: Sequences in an ordered field

```
Let a,b denote sequences.  Let x,y denote reals.

Theorem.  Assume that a[n] = inv(n) for every positive
natural number n.  Then a converges to 0.

Theorem.  Every convergent sequence is bounded.

Theorem.  Assume a converges to x and b converges to y.
Then a +' b converges to x + y.

Theorem.  If a converges to x and a converges to y then x =
y.

Theorem.  Assume a and b are convergent.  If a[n] =< b[n]
for every natural number n then lim a =< lim b.
```

# Example: Sequences in an ordered field

Result:

```
sections 925 - goals 322 - trivial 44 - proved 349 -
equations 150
symbols 3811 - checks 3772 - trivial 3712 - proved 60 -
unfolds 59
parser 00:00.42 - reason 00:00.42 - simplifier 00:00.01 -
prover 00:12.84/00:00.33
total 00:13.70
```

# SAD - Where to go from here?

Identify often used figures of mathematical argument and implement support

- reasoning with/about algebraic structures
- reasoning with/about the syntax of terms and formulae
- extension of computational reasoning
- ...

Conjunction of ForTheL texts remains a problem

- How should a library look?
- How should import and export work?

Can SAD communicate with other proof assistants?