Formalization of the fundamental group in untyped set theory using auto2

Bohua Zhan

Massachusetts Institute of Technology

bzhan@mit.edu

September 29, 2017

Contribution

- Using Isabelle/FOL, we develop mathematics starting from the ZFC axioms, up to the definition of the fundamental group.
- Approx. 13,000 lines of theory files, 3,500 lines of ML code. 5 months of work.
- Need to work with:
  - Algebraic and topological structures.
  - Quotients.
  - Induction (e.g. on natural numbers, finite sets, etc).
  - Arithmetic (e.g. for constructing the real numbers).

Motivation

- Auto2 is a proof automation tool for Isabelle, introduced at ITP 2016.
- In the previous paper, several case studies are given, but they are all fairly short, and the use of auto2 is mixed with the use of other automation tools in Isabelle.
- In the present work, we demonstrate that auto2 can work independently to support formalizations on a relatively large scale.

Why set theory?

- Set theory is the standard foundation for modern mathematics. A system based on set theory can use definitions very close to standard mathematical practice.
- Certain advanced constructions in mathematics are done in a particularly "type-free" way (e.g. algebraic closure of an arbitrary field). Types can get in the way when formalizing such constructions.
- We demonstrate that, with proper automation, it is no more difficult to formalize mathematics in set theory than in type theories.

Comparison to other systems

- Compared to Isabelle/ZF and IsarMathLib:
  - Formalized deeper mathematics.
  - Use auto2 exclusively for proofs. More succinct proof scripts.
- Compared to Mizar:
  - Simple underlying logic. Many constructions added outside the kernel.
  - Emphasis on powerful, extensible automation.

Introduction to Isabelle/FOL + ZFC axioms

- Primitive types: i for sets and o for propositions.
- Function types: $i \to o$, $i \to i$, $(i \to o) \to o$, etc.
- Enough higher-order features to state and use induction rules.
- However, no equality except for types i and o. Any functions that we wish to consider as first-class objects should be defined as set-theoretic functions.
- Similar statement of ZFC axioms as in Isabelle/ZF and IsarMathLib.

Introduction to auto2

- Saturation-based prover for classical logic.
- Independent from existing automation in Isabelle, such as Sledgehammer or the usual Isabelle tactics.
- Proof state consists of a list of items (derived facts, terms, etc), as well as several data structures (e.g. congruence closure of the known equalities).
- *Proof steps* are functions for producing new items from existing ones. They can be as simple as applying a single lemma, or implement more complex proof procedures.

Proof scripts for auto2

- Declarative style: consists solely of intermediate goals with hierarchical structure.
- Compared to Mizar/Isar:
  - No labeling of intermediate goals.
  - No names of tactics.
  - No names of previous lemmas.

Techniques for working with set theory

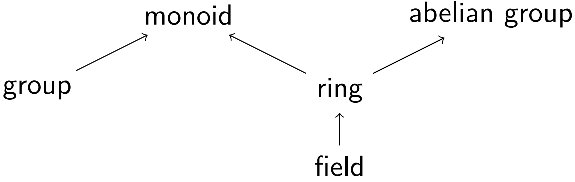
- Abstraction of definitions.
- Properties.
- Well-formed terms and conversions.

Abstraction of definitions

- Many concepts, such as ordered pairs or natural numbers, are represented as sets, but we never make use of their representations except to prove basic facts about these concepts.
- We can abstract away the underlying representation using the following procedure:
  - Step 1: define the concept, add definition as a rewrite rule to auto2.
  - Step 2: prove basic facts, add them as appropriate reasoning rules.
  - Step 3: delete the rewrite rule for the definition from auto2.
- At the end of this procedure, the original definition is effectively hidden away from proof automation, and only the derived facts will be used.

Properties

- Concepts such as *group*, *ring*, and *field*, which may be declared as type-classes in Isabelle/HOL or similar systems, are represented as predicates (terms of type $i \to o$).
- There may be extensive dependencies between such predicates:



- In auto2, we can register any predicate as a *property*. During proof, the *property table* maintains the list of known properties about existing terms. Dependency relations between properties are automatically propagated.

Well-formed terms

- We define a concept of well-formed terms *for use in automation only*.
- For any meta-function, we can register *well-formedness conditions*. These are conditions that should be satisfied by the arguments of the function. For example:

Term	Conditions
$\bigcap A$	$A \neq \emptyset$
$a +_R b$	$a \in \text{carrier}(R), b \in \text{carrier}(R)$
$\text{inv}(R, a)$	$a \in \text{units}(R)$
$\text{subgroup}(G, H)$	$\text{is\_subgroup\_set}(G, H)$
$\text{quotient\_group}(G, H)$	$\text{is\_normal\_subgroup\_set}(G, H)$

- During proof, the *well-form table* maintains the list of known well-formedness conditions of existing terms.

Well-formed conversions

- In an untyped theory, algebraic normalization is more complex, since the relevant rewriting rules have extra conditions.
- E.g.: rule for associativity of addition:

```
"is_abgroup(G) ⟹ x ∈. G ⟹ y ∈. G ⟹ z ∈. G ⟹
  x +_G (y +_G z) = (x +_G y) +_G z"
```

- A *well-formed conversion* takes a term s with well-formedness conditions, and produces an equation $s = t$, together with well-formedness conditions on t. They can be composed just like regular conversions.
- By composing well-formed conversions, one can implement normalization in groups, rings, etc. in a way analogous to that in typed theories.

Examples

- Definition of the fundamental group.
- Rempe-Gillen's challenge.
- Schroeder-Bernstein Theorem.

Definition of the fundamental group

- Given topological space X and a point x on X, the group $\pi_1(X, x)$ is defined on the set of loops based at x modulo path homotopy. The identity element is given by the constant loop at x, and multiplication is given by adjoining paths.
- Formal definition:

```
definition fundamental_group :: "i ⇒ i ⇒ i" ("π₁") where [rewrite]:
  "π₁(X,x) = (let ℛ = loop_space_rel(X,x) in
    Group(loop_classes(X,x), equiv_class(ℛ,const_mor(I,X,x)),
          λf g. equiv_class(ℛ,rep(ℛ,f) ⋆ rep(ℛ,g))))"
```

- Fundamental group is a group:

```
lemma fundamental_group_is_group:
  "is_top_space(X) ⟹ x ∈. X ⟹ is_group(π₁(X,x))"
```

Rempe-Gillen's challenge

Let f be a continuous real-valued function on the real line, such that $f(x) > x$ for all x. Let x_0 be a real number, and define the sequence x_n recursively by $x_{n+1} := f(x_n)$. Then x_n diverges to infinity.

```
lemma rempe_gillen_challenge:
  "real_fun(f) ⟹ continuous(f) ⟹ incr_arg_fun(f) ⟹ x0 ∈. ℝ ⟹
   S = Seq(ℝ,λn. nfold(f,n,x0)) ⟹ ¬upper_bounded(S)"
@proof
  @contradiction
  @have "seq_incr(S)" @with @have "∀n∈.ℕ. S`(n +ℕ 1) ≥ℝ S`n" @end
  @obtain x where "converges_to(S,x)" @then
  @let "T = Seq(ℝ,λn. f`(S`n))" @then
  @have "converges_to(T,f`x)" @then
  @have "converges_to(T,x)" @with
    @have "∀r>ℝ0ℝ. ∃k∈.ℕ. ∀n≥ℕk. ¦T`n -ℝ x¦ℝ <ℝ r" @with
      @obtain "k ∈. ℕ" where "∀n≥ℕk. ¦S`n -ℝ x¦ℝ <ℝ r" @then
      @have "∀n≥ℕk. ¦T`n -ℝ x¦ℝ <ℝ r" @with @have "T`n = S`(n +ℕ 1)" @end @end
  @end
@qed
```

Schroeder-Bernstein Theorem

Given two sets X and Y. If there is an injection f from X to Y and an injection g from Y to X, then there exists a bijection between X and Y.

```
lemma schroeder_bernstein:
  "injective(f) ⟹ injective(g) ⟹ f ∈ X → Y ⟹ g ∈ Y → X ⟹ equipotent(X,Y)"
@proof
  @let "X_A = lfp(X, λW. X - g``(Y - f``W))" @then
  @let "X_B = X - X_A" "Y_A = f``X_A" "Y_B = Y - Y_A" @then
  @have "X - g``Y_B = X_A" @then
  @have "g``Y_B = X_B" @then
  @let "f' = func_restrict_image(func_restrict(f,X_A))" @then
  @let "g' = func_restrict_image(func_restrict(g,Y_B))" @then
  @have "glue_function2(f', inverse(g')) ∈ (X_A ∪ X_B) ≅ (Y_A ∪ Y_B)"
@qed
```

Conclusion

- We created a new library of mathematics based on Isabelle/FOL, showing the feasibility of formalizing advanced mathematics on this logical foundation, and using auto2 exclusively for automation.
- Code available at: https://github.com/bzhan/auto2
- Future work:
  - Still a lot of room for performance improvements.
  - Develop the library in other areas of mathematics.