Inductive types[1]

András Kovács

Eötvös Loránd University, Department of Programming Languages and Compilers

11 Jan 2019

[1]This work was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

Introduction

This talk is a brief introduction to 2018 research on inductive types, by Ambrus Kaposi and myself, with additional contribution from Thorsten Altenkirch and Péter Diviánszky. This has been my main EFOP research topic in the past year.

Related talks and publications:

- Altenkirch, Diviánszky, Kaposi, Kovács: *Constructing inductive-inductive types using a domain-specific type theory*, TYPES 2018
- Kaposi, Kovács: *A Syntax for Higher Inductive-Inductive Types*, FSCD 2018 (award: best paper by junior researchers)
- Kaposi, Kovács, Altenkirch: *Constructing Quotient Inductive-Inductive Types*, POPL 2019

Introduction

Proof by induction is well-known. However, the concept is often left imprecise. Some questions:

- What is exactly an inductive definition?
- When is proof by induction valid?
- How can we get a precise logical statement of induction for an inductive definition?
- What properties should inductive structures have besides having induction principles?

Introduction

Proof by induction is well-known. However, the concept is often left imprecise. Some questions:

- What is exactly an inductive definition?
- When is proof by induction valid?
- How can we get a precise logical statement of induction for an inductive definition?
- What properties should inductive structures have besides having induction principles?

These are general questions of mathematical foundations, and are not necessarily tied to practical formal methods.

Examples of inductive definitions

- Inductive sets: natural numbers, (syntax) trees, lists.
- Inductive relations: typing derivations, reduction in operational semantics, proof trees for logics.

Examples of inductive definitions

- Inductive sets: natural numbers, (syntax) trees, lists.
- Inductive relations: typing derivations, reduction in operational semantics, proof trees for logics.

Induction on natural numbers or lists is obvious, but induction on typing derivations is less so. Just writing down the statement of induction can be challenging for many structures.

Common features

- A collection of sets and families of sets, called sorts.
- A collection of production rules (constructors), each producing elements of some set.
- The assumption that elements of the sorts are precisely those generated by finitely many applications of constructors.

Example with single sort:

```
Nat  : Set
zero : Nat
suc  : Nat → Nat
```

If we allow an infinite number of `suc`-s, `Nat`-induction becomes **false**.

Quotient induction

We may also allow *equations* besides production rules.

Example: a quotient inductive definition of integers.

```
Int     : Set
zero    : Int
suc     : Int → Int
pred    : Int → Int
sucPred : ∀ i → suc (pred i) = i
predSuc : ∀ i → pred (suc i) = i
```

All constructions defined on integers must respect the equations. This is enforced by the `Int`-induction principle.

E. g. we cannot define a function from integers such that the results for `suc (pred i)` and `i` are different.

Quotient induction

Quotient induction is a powerful tool for formalizing lots of mathematics, including

- Real, ordinal, surreal numbers.
- Syntax and semantics of languages, especially those with polymorphic and dependent types.

Quotient induction

Quotient induction is a powerful tool for formalizing lots of mathematics, including

- Real, ordinal, surreal numbers.
- Syntax and semantics of languages, especially those with polymorphic and dependent types.

No current proof assistant supports quotient inductive types.

A key motivation of this research is to build theoretical background for future proof assistants.

(Besides doing pure theory, we're also writing prototype implementations)

Results on quotient induction

In the POPL paper.

- A formal description of valid quotient inductive definitions, as typing contexts in a special-purpose type theory.
- Categorical semantics in rather high detail.
- Showing that every quotient inductive type can be constructed from just one universal quotient inductive type.

Results on higher induction

In the FSCD paper.

Higher induction is used to define spaces generated from n-dimensional constructors (points, paths, surfaces, etc.). It is a broad generalization of quotient induction. It's important in homotopy type theory.

A lot more complicated than quotient induction. Accordingly, our results are significantly weaker for higher induction than for quotient induction.

Future work

- Modest generalizations of current results.
- Write PhD thesis on quotient induction.
- Combine research with setoid type theory (reasearch by INRIA Nantes, Ambrus, Thorsten), implement prototype.

Thank you!