Tree grammars for induction on inductive data types modulo equational theories

Gabriel Ebner, Stefan Hetzl

WAIT 2018
2018-06-28

TU Wien

Introduction

- Main challenge: synthesis of induction formula
- Consider proofs of instances $\varphi(t)$ of $\forall x\, \varphi(x)$
  - similar to the constructive ω-rule, bounded model checking, etc.
- Generalize instance proofs via Herbrand's theorem
  - abstracts from propositional reasoning

Herbrand's theorem

Theorem (special case of Herbrand 1930)

Let $\varphi(x)$ be a quantifier-free first-order formula.

Then $\exists x\, \varphi(x)$ is valid iff there exist terms $t_1, \ldots, t_n$ such that $\varphi(t_1) \vee \cdots \vee \varphi(t_n)$ is a tautology.

- works analogously for $\forall x\, \varphi_1(x), \ldots, \forall x\, \varphi_n(x) \vdash \psi$

Induction-elimination

Theorem (Gentzen 1936)

Let π be a proof of $\forall x\, \varphi(x)$ with induction.

Then there exists a proof π_t of $\varphi(t)$ without induction (or cut).

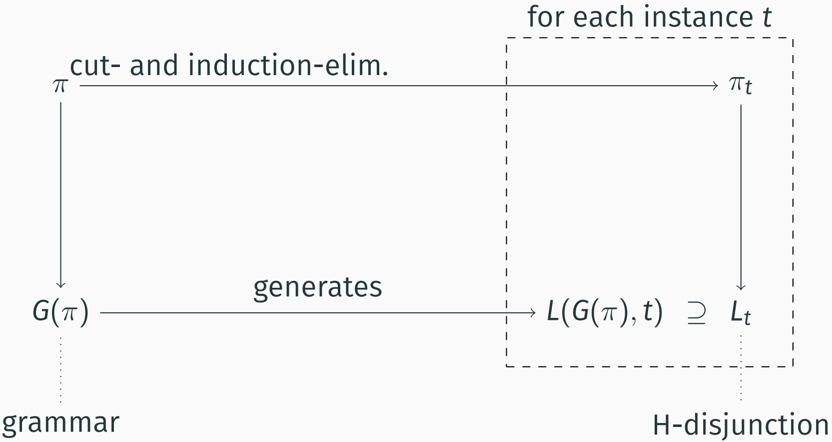
Induction-elimination

Theorem (Gentzen 1936)
Let π be a proof of $\forall x\, \varphi(x)$ with induction.

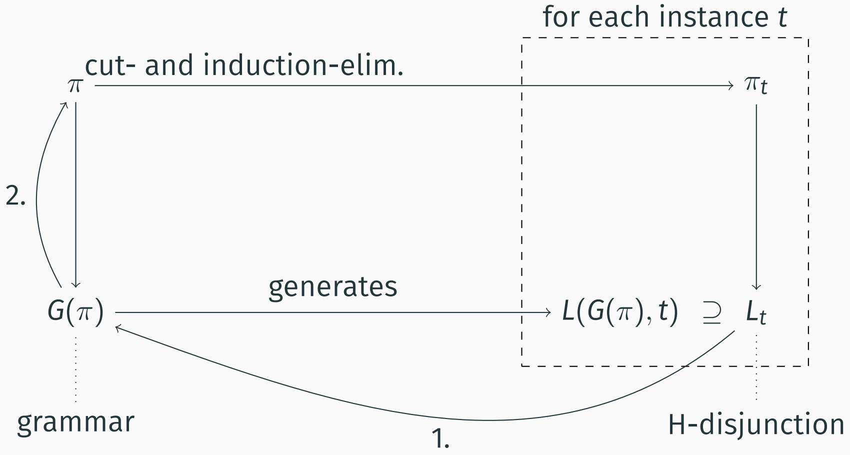
Then there exists a proof π_t of $\varphi(t)$ without induction (or cut).

- t: instance, e.g. $0, s(0), \mathrm{cons}(a, \mathrm{nil})$
- π_t: instance proof

Proofs and grammars (Eberhard, Hetzl 2015)



Proofs and grammars (Eberhard, Hetzl 2015)



Side remark: cut-introduction

- Instead of reconstructing inductions, we can also reconstruct (Π_1-)cuts
- Similar 2-phase approach
  - complete: every generated grammar produces a lemma

→ finds interesting lemmas in practice

New developments

- Implementation
- Inductive data types
- Equational background theories

Equational background theories

- Instance proofs are often irregular

$\rightarrow$ ignore some (formula) instances

- E is a set of (universally quantified) equations
  - e.g. $E = \{x \cdot (y \cdot z) = (x \cdot y) \cdot z\}$

- φ is an E-tautology iff $E \models \varphi$

Inductive data types

- Basic inductive data types
  - not nested, mutual, etc.
- Structural induction

$$\frac{\Gamma \vdash \varphi(\mathrm{nil}) \qquad \Gamma, \varphi(y) \vdash \varphi(\mathrm{cons}(x, y))}{\Gamma \vdash \varphi(t)}$$

Simple induction proofs

- One universally quantified induction
- But different formula
  - (ψ is prenex and universally quantified)

$$\dfrac{\dfrac{\dfrac{\begin{matrix}(\pi_i)\\ \Gamma_i, \psi(\alpha, \nu_{i,j}, \bar{t}), \cdots \vdash \psi(\alpha, c_i(\overline{\nu_i}), \bar{\gamma})\end{matrix}}{\Gamma, \forall \bar{y}\, \psi(\alpha, \nu_{i,j}, \bar{y}), \cdots \vdash \forall \bar{y}\, \psi(\alpha, c_i(\overline{\nu_i}), \bar{y})} \quad \cdots}{\Gamma \vdash \forall \bar{y}\, \psi(\alpha, \alpha, \bar{y})}\,\mathrm{ind}_\rho \quad \dfrac{\begin{matrix}(\pi_c)\\ \Gamma_c, \psi(\alpha, \alpha, \bar{u}), \cdots \vdash \varphi(\alpha)\end{matrix}}{\Gamma, \forall \bar{y}\, \psi(\alpha, \alpha, \bar{y}) \vdash \varphi(\alpha)}}{\Gamma \vdash \varphi(\alpha)}\,\mathrm{cut}}{\Gamma \vdash \forall x\, \varphi(x)}$$

Induction grammar

Definition

Induction grammar is a tuple $G = (\tau, \alpha, (\overline{\nu}_c)_c, \overline{\gamma}, P)$ with productions P of the form:

- $\tau \rightarrow t[\alpha, \overline{\nu}_c, \overline{\gamma}]$
- $\overline{\gamma} \rightarrow \bar{t}[\alpha, \overline{\nu}_c, \overline{\gamma}]$

Induction grammar

Definition

$G(\pi)$ is induction grammar for simple induction proof π

$\rightarrow$ describes quantifier instances

Definition

$L(G, t)$ is the (finite) language of G (t constructor term)

Theorem

$L(G(\pi), t)$ is E-tautological for all t

Example

$$\forall x\,(s(0)\cdot x = x \wedge x\cdot s(0) = x), \tag{f_1}$$

$$\forall x \forall y \forall z\, x\cdot(y\cdot z) = (x\cdot y)\cdot z, \tag{f_2}$$

$$\mathrm{fact}(0) = s(0), \tag{f_3}$$

$$\forall x\, \mathrm{fact}(s(x)) = s(x)\cdot \mathrm{fact}(x), \tag{f_4}$$

$$\forall y\, \mathrm{qfact}(y, 0) = y, \tag{f_5}$$

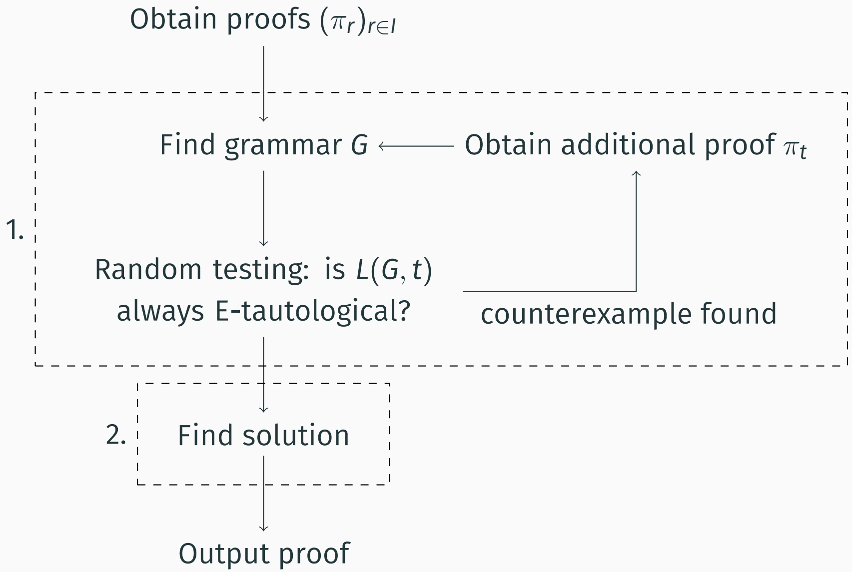
$$\forall x \forall y\, \mathrm{qfact}(y, s(x)) = \mathrm{qfact}(y\cdot s(x), x) \tag{f_6}$$

$$\vdash \forall x\, \mathrm{qfact}(s(0), x) = \mathrm{fact}(x) \tag{goal}$$

$$\tau \to f_3 \mid f_4(\nu) \mid f_5(\gamma) \mid f_6(\nu, \gamma)$$

$$\gamma \to \gamma\cdot s(\nu) \mid s(0)$$

Algorithm overview



Grammar finding

- Given finite collection $t \mapsto L_t$
    - L_t represents a Herbrand disjunction
- Want G such that $L(G, t) \supseteq L_t$

- Find G with minimal number of productions
- using a MaxSAT solver (see also Eberhard, E, Hetzl 2017)

Induced Boolean unification problem

- Induction grammar induces $\mathrm{BUP}_G(X)$
  - $\Gamma_1, \bigwedge_l \bigwedge X(\alpha, \nu_{1,l}, \bar{t}) \vdash X(\alpha, c_1(\overline{\nu_1}), \overline{\gamma})$
  - ...
  - $\Gamma_n, \bigwedge_l \bigwedge X(\alpha, \nu_{n,l}, \bar{t}) \vdash X(\alpha, c_n(\overline{\nu_n}), \overline{\gamma})$
  - $\Gamma_c, \bigwedge X(\alpha, \alpha, \bar{t}) \vdash \varphi(\alpha)$
- There exists simple induction proof with grammar G iff there exists quantifier-free φ s.t. $\mathrm{BUP}_G(\varphi)$ E-tautology

$\rightarrow$ Find quantifier-free X such that all sequents are E-tautological

- even for quantified induction formulas

BUP example

- $\mathrm{qfact}(\gamma, \mathsf{0}) = \gamma, \mathrm{fact}(\mathsf{0}) = \mathsf{s}(\mathsf{0}), \top \vdash X(\alpha, \mathsf{0}, \gamma)$
- $\mathrm{fact}(\mathsf{0}) = \mathsf{s}(\mathsf{0}), \mathrm{fact}(\mathsf{s}(\nu)) = \mathsf{s}(\nu) \cdot \mathrm{fact}(\nu),$
  $\mathrm{qfact}(\gamma, \mathsf{0}) = \gamma, \mathrm{qfact}(\gamma, \mathsf{s}(\nu)) = \mathrm{qfact}(\gamma \cdot \mathsf{s}(\nu), \nu),$
  $X(\alpha, \nu, \mathsf{s}(\mathsf{0})) \wedge X(\alpha, \nu, \gamma \cdot \mathsf{s}(\nu)) \vdash X(\alpha, \mathsf{s}(\nu), \gamma)$
- $\mathrm{fact}(\mathsf{0}) = \mathsf{s}(\mathsf{0}), X(\alpha, \alpha, \mathsf{s}(\mathsf{0})) \vdash \mathrm{qfact}(\mathsf{s}(\mathsf{0}), \alpha) = \mathrm{fact}(\alpha)$

BUP example

- $\text{qfact}(\gamma, 0) = \gamma, \text{fact}(0) = s(0), \top \vdash X(\alpha, 0, \gamma)$
- $\text{fact}(0) = s(0), \text{fact}(s(\nu)) = s(\nu) \cdot \text{fact}(\nu),$
  $\text{qfact}(\gamma, 0) = \gamma, \text{qfact}(\gamma, s(\nu)) = \text{qfact}(\gamma \cdot s(\nu), \nu),$
  $X(\alpha, \nu, s(0)) \wedge X(\alpha, \nu, \gamma \cdot s(\nu)) \vdash X(\alpha, s(\nu), \gamma)$
- $\text{fact}(0) = s(0), X(\alpha, \alpha, s(0)) \vdash \text{qfact}(s(0), \alpha) = \text{fact}(\alpha)$

Solution: $X = \lambda\alpha\lambda\nu\lambda\gamma\,(\text{qfact}(\gamma, \nu) = \gamma \cdot \text{fact}(\nu))$

Canonical formula

- Canonical formula C_t for t instance
  - Simplest case $C_{s(s(0))} = \Gamma_0 \wedge \Gamma_1[\nu \backslash 0] \wedge \Gamma_1[\nu \backslash s(0)]$
- Implies any other solution
  - $C_t \rightarrow \varphi(\alpha, t, \overline{\gamma})$

$\rightarrow$ Solution finding algorithm

1. Compute C_t
2. Enumerate consequences
   - e.g. using forgetful resolution $(a \rightarrow b) \wedge (b \rightarrow c) \rightsquigarrow (a \rightarrow c)$
3. Replace some occurrences of t by ν
4. Check if it is a solution

Undecidability of BUP solution

- Solvability of BUP is undecidable (Eberhard, Hetzl, Weller 2015)
- $L(G, t)$ E-tautological for all $t \Rightarrow$ BUP solvable?
  - unfortunately no

$\rightarrow$ solvability depends on the input proofs

Implementation

- Prototype implementation
- GAPT: General Architecture for Proof Theory
- `https://github.com/gapt/gapt`
- Native support for TIP format

Evaluation on TIP

- Solves about 22 problems out of the box
  - Bit more with manual options
- All with quantifier-free induction formula
- Probably due to lack of regularity in proofs

Reconstruction success

- Does the method work with regular sequences of proofs?
- Tested 52 simple induction proofs
- We can always find a grammar.
- Reconstruction works for 43 proofs.

Case study: schematic CERES

- Analysis of proofs with induction (Cerna, Leitsch, Lolic; ongoing work)
- Requires automatic inductive proof as intermediate step
- Complex induction invariants

$$
\begin{aligned}
&(Omega(\nu) \to E(o, f(S(a)))) \\
\wedge\ &(Omega(\nu) \to E(o, f(a))) \\
\wedge\ &(Omega(\nu) \to Phi(o)) \\
\wedge\ &\neg(Phi(s(\nu)) \wedge Phi(\nu) \wedge Omega(s(\nu)))
\end{aligned}
$$

(automatically found)

Introduction

Proofs and tree grammars

Inductive proving using tree grammars

Evaluation

Conclusion

Future work

- Modify provers to produce more regular proofs
    - e.g. innermost vs. outermost rewriting

- Regularize existing proofs?

- Improve solution finding phase

  $\rightarrow$ constrained Horn clause solvers

Conclusion

- Not yet sufficient for TIP problems
- Alternative challenge:
  - Instead of finding induction formulas,
    find regular sequences of Herbrand disjunctions