

Computational complexity of grammars for proofs with induction

Gabriel Ebner

First International Workshop on
Proof Theory for Automated Deduction,
Automated Deduction for Proof Theory

2019-10-24

Technische Universität Wien / Vrije Universiteit Amsterdam

Introduction

Grammars

Complexity of decision problems

More tractable subclasses

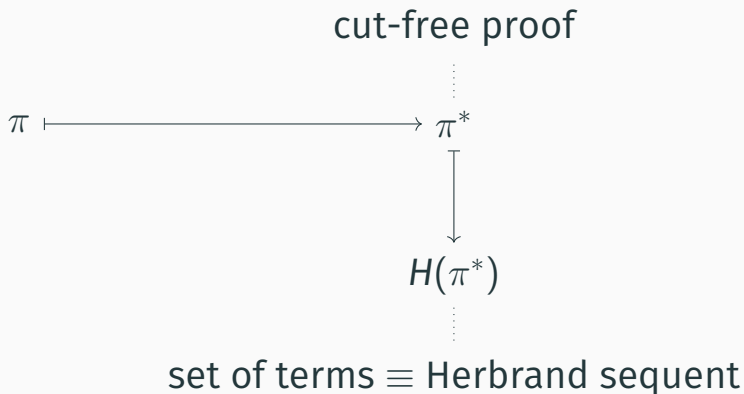
Back to proofs

Conclusion

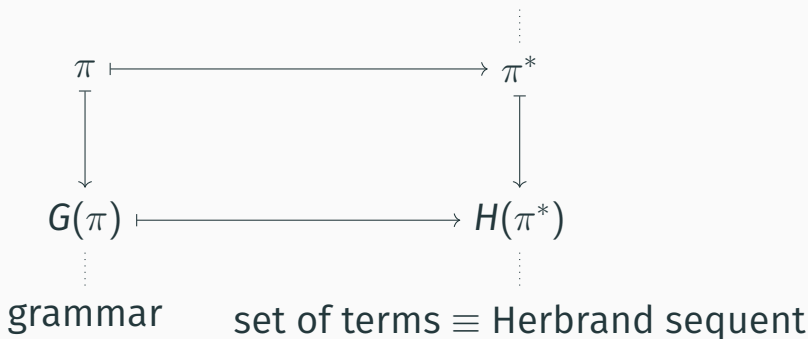
Theorem (special case of Herbrand 1930)

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

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

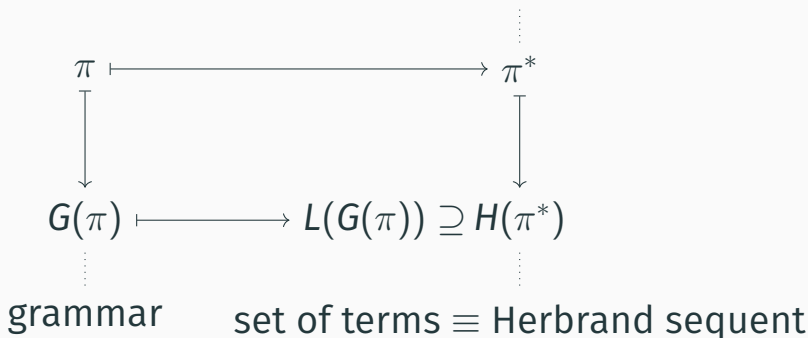


cut-free proof



- Assign (tree) grammar $G(\pi)$ to proof π such that the language $L(G(\pi))$ is a Herbrand sequent

cut-free proof



- Assign (tree) grammar $G(\pi)$ to proof π such that the language $L(G(\pi))$ is a Herbrand sequent

Applications

- Invert cut-elimination: given a cut-free proof π , find a proof π' with cuts.
 1. Find grammar G such that $L(G) \supseteq H(\pi)$
 2. Compute cut-formulas
 - Lemma generation
 - Invariants for inductive proofs
- Working with grammars reduces bureaucracy:
 - e.g. prove uncompressibility result for grammars, then lift to proofs (Eberhard, Hetzl 2018)

Introduction

Grammars

Complexity of decision problems

More tractable subclasses

Back to proofs

Conclusion

Vectorial totally rigid acyclic tree grammars.

Grammars for proofs with purely universally quantified cuts.

- Start symbol: A
- Nonterminal vectors: $A, \bar{B}, \bar{C}, \bar{D}, \dots$
where $\bar{B} = (B_1, \dots, B_n)$, etc.
- (Acyclic) productions: $\bar{B} \rightarrow \bar{t}[\bar{C}, \bar{D}, \dots]$

Rigid derivations: $A[A \setminus t_1][\bar{B} \setminus \bar{t}_2][\bar{C} \setminus \bar{t}_3] \dots$

(Finite!) language $L(G)$ consists of all derivable terms

VTRATG example

$$A \rightarrow f(B_1, B_1, B_2) \mid g(B_1, B_1, B_2)$$

$$\bar{B} \rightarrow (c, e) \mid (d, f)$$

VTRATG example

$$A \rightarrow f(B_1, B_1, B_2) \mid g(B_1, B_1, B_2)$$

$$\bar{B} \rightarrow (c, e) \mid (d, f)$$

$$L(G) = \{f(c, c, e), f(d, d, f), g(c, c, e), g(d, d, f)\}$$

Cut-reduction to language generation

$L(G(\pi))$ contains the formulas in a Herbrand sequent of π

$G(\pi)$ consists of:

- Nonterminals: eigenvariables from cuts + start symbol A
- Productions $\alpha \rightarrow t$ for weak quantifier inferences on cut formulas:

$$\frac{\frac{\vdash \varphi(\alpha)}{\vdash \forall x \varphi(x)} \forall_r \quad \frac{\frac{\varphi(t) \vdash}{\vdash} \forall_l \quad \vdots}{\forall x \varphi(x) \vdash}}{\text{cut}}$$

- Productions $A \rightarrow \varphi(t)$ for instances of formulas end-sequent.

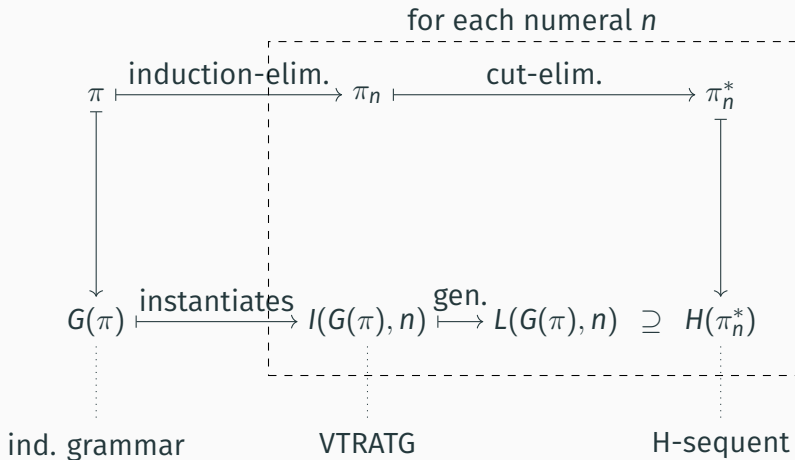
Simple induction proofs (Eberhard, Hetzl 2015)

$$\frac{\frac{\frac{(\pi_1)}{\Gamma \vdash \forall x \psi(0, \bar{x})} \quad \frac{(\pi_2)}{\Gamma, \forall x \psi(s(\nu), \bar{x}) \vdash \forall \bar{x} \psi(\nu, \bar{x})}}{\Gamma \vdash \forall \bar{x} \psi(\alpha, \bar{x})} \text{ind}}{\Gamma \vdash \varphi(\alpha)} \quad \frac{(\pi_3)}{\Gamma, \forall \bar{x} \psi(\alpha, \bar{x}) \vdash \varphi(\alpha)}}{\Gamma \vdash \varphi(\alpha)}$$

where π_1, π_2, π_3 are cut-free
and ψ, φ are quantifier-free

→ study induced Herbrand sequents of $\Gamma \vdash \varphi(n)$ for numerals n .

Grammar assignment to simple induction proofs (Eberhard, Hetzl 2015)



Induction grammars

- Two kinds of (cyclic!) productions:
 - $\tau \rightarrow t[\bar{\gamma}, \alpha, \nu]$
 - $\bar{\gamma} \rightarrow t[\bar{\gamma}, \alpha, \nu]$
- Instantiation: for each numeral n , set $L(G, n) = L(I(G, n))$ for VTRATG $I(G, n)$:
 - Nonterminals $\tau, \bar{\gamma}_0, \dots, \bar{\gamma}_n$.
 - $\tau \rightarrow t[\alpha, \nu, \bar{\gamma}] \rightsquigarrow \tau \rightarrow t[n, k, \bar{\gamma}_{s(k)}]$ for $s(k) < n$
 - $\bar{\gamma} \rightarrow \bar{t}[\alpha] \rightsquigarrow \bar{\gamma}_k \rightarrow \bar{t}[n]$ for $k < n$
 - $\bar{\gamma} \rightarrow \bar{t}[\alpha, \nu, \bar{\gamma}] \rightsquigarrow \bar{\gamma}_k \rightarrow \bar{t}[n, k, \bar{\gamma}_{s(k)}]$ for $s(k) < n$
 - (corresponds to unrolling of induction inference)

Induction grammar example

$$\tau \rightarrow r(\gamma_1, \gamma_1) \qquad \bar{\gamma} \rightarrow (f(\gamma_2), g(\gamma_1)) \mid (c, d)$$

Instantiates to $I(G, 2)$:

$$\begin{aligned} \tau &\rightarrow r(\gamma_{0,1}, \gamma_{0,1}) \mid r(\gamma_{1,1}, \gamma_{1,1}) \mid r(\gamma_{2,1}, \gamma_{2,1}) \\ (\gamma_{0,1}, \gamma_{0,2}) &\rightarrow (f(\gamma_{1,2}), g(\gamma_{1,1})) \mid (c, d) \\ (\gamma_{1,1}, \gamma_{1,2}) &\rightarrow (f(\gamma_{2,2}), g(\gamma_{2,1})) \mid (c, d) \\ (\gamma_{2,1}, \gamma_{2,2}) &\rightarrow (c, d) \end{aligned}$$

And $L(G, 2) = \{r(f(g(c)), f(g(c))), r(f(d), f(d)), r(c, c)\}$

Introduction

Grammars

Complexity of decision problems

More tractable subclasses

Back to proofs

Conclusion

Minimal cover

Problem (TRATG-COVER)

Input: set of terms T and a number k .

Output: is there a TRATG G with at most k productions such that $L(G) \supseteq T$?

Surprisingly hard. We only know that it is in NP.

Decision problems on VTRATGs (Eberhard, E, Hetzl 2018)

VTRATG-MEMBERSHIP:	$t \in L(G)$	NP-complete
VTRATG-EMPTINESS:	$L(G) = \emptyset$	coNP-complete
VTRATG-CONTAINMENT:	$L(G_1) \subseteq L(G_2)$	Π_2^P -complete
VTRATG-DISJOINTNESS:	$L(G_1) \cap L(G_2) = \emptyset$	coNP-complete
VTRATG-EQUIVALENCE:	$L(G_1) = L(G_2)$	Π_2^P -complete

Decision problems on induction grammars

IND-MEMBERSHIP:	$t \in L(G, n)$	NP-complete
IND-EMPTINESS:	$\forall n L(G, n) = \emptyset$	PSPACE-complete
IND-CONTAINMENT:	$\forall n L(G_1, n) \subseteq L(G_2, n)$	undecidable
IND-DISJOINTNESS:	$\forall n L(G_1, n) \cap L(G_2, n) = \emptyset$	undecidable
IND-EQUIVALENCE:	$\forall n L(G_1, n) = L(G_2, n)$	undecidable

Post Correspondence Problem

Problem (PCP)

Input: two finite lists of words w_1, \dots, w_n and v_1, \dots, v_n

Output: is there a sequence of indices i_1, \dots, i_k with $k > 0$ such that $w_{i_1} \dots w_{i_k} = v_{i_1} \dots v_{i_k}$?

Undecidable (Post 1946).

Disjointness

Theorem

IND-DISJOINTNESS is undecidable.

Proof.

Reduce PCP to IND-DISJOINTNESS.

Construct two induction grammars Image_P and $\text{Equal}_{\Sigma, l}$:

- Image_P generates all pairs $(w_{i_1} \cdots w_{i_l}, v_{i_1} \cdots v_{i_l})$
- $\text{Equal}_{\Sigma, l}$ generates all pairs (w, w)

A word $a_1 a_2 \dots a_n$ is encoded as a unary term $a_1(a_2(\dots a_n(\epsilon)))$.

Then the PCP instance has *no* solution iff:

$$\forall i \quad L(\text{Image}_P, i) \cap L(\text{Equal}_{\Sigma, l}, i) = \emptyset \quad \square$$

Disjointness of induction grammars

Let $w = a_1 a_2 \dots a_k$, then $w \cdot \gamma = a_1(a_2(\dots a_k(\gamma)))$.

Definition

The induction grammar $\text{Equal}_{\Sigma, l}$ has the following productions:

$$\tau \rightarrow r(\gamma, \gamma)$$

$$\gamma \rightarrow w \cdot \gamma \mid w \quad \text{where } |w| \leq l$$

Lemma

$$L(\text{Equal}_{\Sigma, l}, k) = \{r(w, w) \mid w \in \Sigma^*, |w| \leq l(k + 1)\}$$

Disjointness of induction grammars

Definition

The induction grammar Image_p has the following productions:

$$\tau \rightarrow r(\gamma_1, \gamma_2)$$

$$(\gamma_1, \gamma_2) \rightarrow (w_1 \cdot \gamma_1, v_1 \cdot \gamma_2) \mid \cdots \mid (w_n \cdot \gamma_1, v_n \cdot \gamma_2)$$

$$(\gamma_1, \gamma_2) \rightarrow (w_1, v_1) \mid \cdots \mid (w_n, v_n)$$

Lemma

$$L(\text{Image}_p, k) = \{r(w_{i_1} \cdots w_{i_l}, v_{i_1} \cdots v_{i_l}) \mid 1 \leq l \leq k + 1\}$$

Containment of induction grammars

Theorem

IND-CONTAINMENT is undecidable.

Proof.

Similar to IND-DISJOINTNESS.

Construct two induction grammars Image_p and $\text{Diff}_{\Sigma,l}$:

- Image_p as before
- $\text{Diff}_{\Sigma,l}$ generates all pairs of different words

Then the PCP instance has *no* solution iff:

$$\forall i \quad L(\text{Image}_p, i) \subseteq L(\text{Diff}_{\Sigma,l}, i)$$

□

Containment of induction grammars

Definition

The induction grammar $\text{Diff}_{\Sigma, l}$ has the following productions:

$$\tau \rightarrow r(\gamma_1, \gamma_2)$$

$$\bar{\gamma} \rightarrow (t \cdot \gamma_1, u \cdot \gamma_2, v \cdot \gamma_3, w \cdot \gamma_4) \quad \text{where } |t| = |u| \leq l \wedge \max(|v|, |w|) \leq l$$

$$\bar{\gamma} \rightarrow (t \cdot \gamma_3, u \cdot \gamma_4, v \cdot \gamma_3, w \cdot \gamma_4) \quad \text{where } |t| = |u| \leq l \wedge \max(|v|, |w|) \leq l \wedge t \neq u$$

$$\bar{\gamma} \rightarrow (t, u, v, w) \quad \text{where } \max(|t|, |u|, |v|, |w|) \leq l \wedge t \neq u$$

where $t, u, v, w \in \Sigma^*$ and $\bar{\gamma} = (\gamma_1, \gamma_2, \gamma_3, \gamma_4)$

Lemma

$$L(\text{Diff}_{\Sigma, l}, k) = \{r(v, w) \mid v \neq w \in \Sigma^* \wedge \max(|v|, |w|) \leq l(k + 1)\}$$

Introduction

Grammars

Complexity of decision problems

More tractable subclasses

Back to proofs

Conclusion

Dependency graphs of VTRATGs

Hardness results on VTRATGs require complicated grammars.
Typical grammars (such as $I(G, n)$) are much simpler.

Dependency graphs of VTRATGs

Hardness results on VTRATGs require complicated grammars.
Typical grammars (such as $I(G, n)$) are much simpler.

This VTRATG G is almost “linear”:

$$A \rightarrow f(B) \quad B \rightarrow f(C) \quad C \rightarrow g(D) \quad D \rightarrow c$$

Dependency graphs of VTRATGs

Hardness results on VTRATGs require complicated grammars.
Typical grammars (such as $I(G, n)$) are much simpler.

This VTRATG G is almost “linear”:

$$A \rightarrow f(B) \quad B \rightarrow f(C) \quad C \rightarrow g(D) \quad D \rightarrow c$$

Assign dependency graph $D(G)$:

$$A - B - C - D$$

Dependency graphs of VTRATGs

Hardness results on VTRATGs require complicated grammars. Typical grammars (such as $I(G, n)$) are much simpler.

This VTRATG G is almost “linear”:

$$A \rightarrow f(B) \quad B \rightarrow f(C) \quad C \rightarrow g(D) \quad D \rightarrow c$$

Assign dependency graph $D(G)$:

$$A - B - C - D$$

The treewidth $\text{tw}(D(G))$ measures how close it is to a tree:

- Let G be a connected graph with at least two vertices, then $\text{tw}(G) = 1$ iff G is a tree

Dependency graphs of VTRATGs

Hardness results on VTRATGs require complicated grammars.
Typical grammars (such as $I(G, n)$) are much simpler.

This VTRATG G is almost “linear”:

$$A \rightarrow f(B) \quad B \rightarrow f(C) \quad C \rightarrow g(D) \quad D \rightarrow c$$

Assign dependency graph $D(G)$:

$$A - B - C - D$$

The treewidth $\text{tw}(D(G))$ measures how close it is to a tree:

- Let G be a connected graph with at least two vertices, then $\text{tw}(G) = 1$ iff G is a tree

We have $\text{tw}(D(I(G, n))) \leq 2|\bar{\gamma}|!$

Treewidth-bounded dependency graphs

($\text{tw} \leq k$)-MEMBERSHIP:	$t \in L(G)$	P
($\text{tw} \leq k$)-EMPTYNESS:	$L(G) = \emptyset$	P
($\text{tw} \leq k$)-CONTAINMENT:	$L(G_1) \subseteq L(G_2)$	coNP-complete
($\text{tw} \leq k$)-DISJOINTNESS:	$L(G_1) \cap L(G_2) = \emptyset$	coNP-complete
($\text{tw} \leq k$)-EQUIVALENCE:	$L(G_1) = L(G_2)$	coNP-complete

These complexity results apply to $I(G, n)$ since we have $\text{tw}(D(I(G, n))) \leq 2|\bar{\gamma}|$.

Induction grammars with bounded $|\bar{\gamma}|$

$(\bar{\gamma} \leq k)$ -IND-MEMBERSHIP:	$t \in L(G)$	P
$(\bar{\gamma} \leq k)$ -IND-EMPTINESS:	$\forall n L(G, n) = \emptyset$	P
$(\bar{\gamma} \leq k)$ -IND-CONTAINMENT:	$\forall n L(G_1, n) \subseteq L(G_2, n)$	undec.
$(\bar{\gamma} \leq k)$ -IND-DISJOINTNESS:	$\forall n L(G_1, n) \cap L(G_2, n) = \emptyset$	undec.
$(\bar{\gamma} \leq k)$ -IND-EQUIVALENCE:	$\forall n L(G_1, n) = L(G_2, n)$	undec.

Introduction

Grammars

Complexity of decision problems

More tractable subclasses

Back to proofs

Conclusion

Problems on proofs

- Complexity results also transfer to proofs.
- Want to find simple induction proofs $\pi(G)$ such that e.g.:

$$L(G(\pi(G_1)), n) \subseteq L(G(\pi(G_2)), n) \leftrightarrow H(\pi(G_1)_n^*) \subseteq H(\pi(G_2)_n^*)$$

- Main technical challenge: weakening inferences.
 - in general $L(G(\pi), n) \supset H(\pi_n^*)!$

Theorem (Hetzl, Straßburger 2012)

- For every Gentzen cut-reduction sequence $\pi \rightsquigarrow \pi'$, we have $L(G(\pi)) \supseteq L(G(\pi'))$.
- If we did not perform grade reduction on weakenings, then $L(G(\pi)) = L(G(\pi'))$.

Let \rightsquigarrow^{ne} be the *non-erasing* Gentzen cut-reduction relation, i.e. where we do not reduce weakenings.

We can still define $H(\cdot)$ on \rightsquigarrow^{ne} -NFs.

Decision problems on proofs

Problem (SIP-CONTAINMENT).

Input: simple induction proofs π, π' .

Let $\pi_n^*, \pi_n'^*$ be \rightsquigarrow^{ne} -NFs such that $\pi_n \rightsquigarrow^{ne} \pi_n^*$ and $\pi_n' \rightsquigarrow^{ne} \pi_n'^*$.

Output: is $H(\pi_n^*) \subseteq H(\pi_n'^*)$ for all n ?

Theorem

SIP-CONTAINMENT is undecidable.

Introduction

Grammars

Complexity of decision problems

More tractable subclasses

Back to proofs

Conclusion

Conclusion

- Decision problems on induction grammars are generally infeasible.
 - Even restricting the size of the vectors.
- **Open problem:** how complex is IND-COVER?
(or TRATG-COVER, resp.?)

Given a finite family of sets of terms $(L_n)_{n \in I}$ and $K \geq 0$, is there an induction grammar G with at most K productions such that $L(G, n) \supseteq L_n$ for all n ?