Applying Tree Languages in Proof Theory

Stefan Hetzl
Institute of Discrete Mathematics and Geometry
Vienna University of Technology

LATA 2012
Language and Automata Theory and Applications

A Coruña, Spain

March 6, 2012

Motivation

- Proof theory
  - Hilbert's Programme, Foundations of Mathematics, $\sim$ 1920s
  - Study mathematical proof as formal objects (i.e. strings)

Motivation

- Proof theory
  - Hilbert's Programme, Foundations of Mathematics, $\sim$ 1920s
  - Study mathematical proof as formal objects (i.e. strings)
- Proof mining
  - Extract concrete information from abstract proofs
  - Example: proof of $\exists x\,(x = f(x))$. Find such x, a "witness".

Motivation

- Proof theory
  - Hilbert's Programme, Foundations of Mathematics, $\sim$ 1920s
  - Study mathematical proof as formal objects (i.e. strings)
- Proof mining
  - Extract concrete information from abstract proofs
  - Example: proof of $\exists x\,(x = f(x))$. Find such x, a "witness".
  - **Theorem.** There are $x, y \in \mathbb{R} \setminus \mathbb{Q}$ s.t. $x^y \in \mathbb{Q}$.
    Proof. If $\sqrt{2}^{\sqrt{2}} \in \mathbb{Q}$, let $x = y = \sqrt{2}$ and we are done as $\sqrt{2} \in \mathbb{R} \setminus \mathbb{Q}$. Otherwise $\sqrt{2}^{\sqrt{2}} \in \mathbb{R} \setminus \mathbb{Q}$, let $x = \sqrt{2}^{\sqrt{2}}, y = \sqrt{2}$ and observe $x^y = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = \sqrt{2}^2 = 2 \in \mathbb{Q}$. $\square$
  - In general: a finite set of witnesses

Motivation

- Proof theory
  - Hilbert's Programme, Foundations of Mathematics, $\sim$ 1920s
  - Study mathematical proof as formal objects (i.e. strings)
- Proof mining
  - Extract concrete information from abstract proofs
  - Example: proof of $\exists x\,(x = f(x))$. Find such x, a "witness".
  - **Theorem.** There are $x, y \in \mathbb{R} \setminus \mathbb{Q}$ s.t. $x^y \in \mathbb{Q}$.
    Proof. If $\sqrt{2}^{\sqrt{2}} \in \mathbb{Q}$, let $x = y = \sqrt{2}$ and we are done as $\sqrt{2} \in \mathbb{R} \setminus \mathbb{Q}$. Otherwise $\sqrt{2}^{\sqrt{2}} \in \mathbb{R} \setminus \mathbb{Q}$, let $x = \sqrt{2}^{\sqrt{2}}, y = \sqrt{2}$ and observe $x^y = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = \sqrt{2}^2 = 2 \in \mathbb{Q}$. $\square$
  - In general: a finite set of witnesses, i.e. a finite tree language!

Outline

- Proof mining: cut-elimination
- Rigid tree languages
- From proofs to grammars
- From grammars to proofs

Cut-Elimination

- *Cut*: formalisation of the use of a lemma

$$\frac{T \vdash A \quad T, A \vdash B}{T \vdash B}\ \text{cut}$$

- *Cut-elimination*: stepwise transformation of proof
- *Cut-free proof*: possible to read of witnesses
- *Witnesses for* $T \vdash \exists x\, A$: $t_1, \ldots, t_n$ s.t. $T \vdash \bigvee_{i=1}^{n} A[x \backslash t_i]$.

Cut-Elimination

- *Cut*: formalisation of the use of a lemma

$$\frac{T \vdash A \quad T, A \vdash B}{T \vdash B}\ \text{cut}$$

- *Cut-elimination*: stepwise transformation of proof
- *Cut-free proof*: possible to read of witnesses
- *Witnesses for* $T \vdash \exists x\, A$: $t_1, \ldots, t_n$ s.t. $T \vdash \bigvee_{i=1}^{n} A[x \backslash t_i]$.

- Basic idea of this talk:

$$\begin{array}{ccc} \text{proof } \pi \text{ with cuts} & \longrightarrow^{\text{cut-elimination}} & \text{cut-free proof } \pi^* \\ \downarrow & & \downarrow \\ \text{grammar } \mathrm{G}(\pi) & \longrightarrow^{\text{defines}} & \begin{array}{c}\text{witnesses of } \pi^* \\ = \mathrm{L}(\mathrm{G}(\pi))\end{array} \end{array}$$

Outline

- $\surd$ Proof mining: cut-elimination
- Rigid tree languages
- From proofs to grammars
- From grammars to proofs

Tree Automata with Equality Contraints

- Local equality contraints, e.g. $f(q_1, q_1) \overset{1=2}{\rightarrow} q_2$
- Global equality contraints via states, e.g. TAGED [Filiot, Talbot, Tison '07]
- Rigid tree automata [Jacquemard, Clay, Vacher '09]
  - subclass of TAGED
  - Rigid tree automaton $\langle Q, R, F, \Delta \rangle$ where $R \subseteq Q$
  - *Rigidity condition* on run $r : \mathrm{Pos}(t) \to Q$:
    $\forall p_1, p_2 \in \mathrm{Pos}(t)$ with $r(p_1) = r(p_2) \in R$: $t|_{p_1} = t|_{p_2}$.
  - $\mathrm{L}(\mathcal{A})$ = all terms which have runs satisfying rigidity condition

Rigid Tree Grammars

- Rigid tree grammar $\langle \alpha, N, R, \Sigma, P \rangle$ where $R \subseteq N$ *rigid*
- *Rigidity condition* on derivation:
  if two productions with $\beta \in R$ as left hand side are applied at positions p_1, p_2, then $t|_{p_1} = t|_{p_2}$.
- Example: $\alpha \to f(\beta, \beta), \beta \to g(\gamma), \gamma \to a \mid g(\gamma)$ with $R = \{\beta\}$ has $L = \{f(g^n(a), g^n(a)) \mid n \geq 1\}$.

Rigid Tree Grammars

- Rigid tree grammar $\langle \alpha, N, R, \Sigma, P \rangle$ where $R \subseteq N$ *rigid*
- *Rigidity condition* on derivation:
  if two productions with $\beta \in R$ as left hand side are applied at positions p_1, p_2, then $t|_{p_1} = t|_{p_2}$.
- Example: $\alpha \to f(\beta, \beta), \beta \to g(\gamma), \gamma \to a \mid g(\gamma)$ with $R = \{\beta\}$ has $L = \{f(g^n(a), g^n(a)) \mid n \geq 1\}$.
- **Theorem.** L language of a rigid tree grammar iff L language of rigid tree automaton.

Rigid Tree Grammars

- Rigid tree grammar $\langle \alpha, N, R, \Sigma, P \rangle$ where $R \subseteq N$ *rigid*
- *Rigidity condition* on derivation:
  if two productions with $\beta \in R$ as left hand side are applied at positions p_1, p_2, then $t|_{p_1} = t|_{p_2}$.
- Example: $\alpha \to f(\beta, \beta), \beta \to g(\gamma), \gamma \to a \mid g(\gamma)$ with $R = \{\beta\}$ has $L = \{f(g^n(a), g^n(a)) \mid n \geq 1\}$.
- **Theorem.** L language of a rigid tree grammar iff L language of rigid tree automaton.
- **Definition.** A grammer is called *totally rigid* if $N = R$.
- **Definition.** A grammar is called acyclic if there is no derivation $\beta \to t$ with $\beta \in \mathrm{V}(t)$
- this paper: totally rigid acyclic tree grammars **(!)**

Outline

- $\surd$ Proof mining: cut-elimination
- $\surd$ Rigid tree languages
- From proofs to grammars
- From grammars to proofs

From Proofs to Grammars

- **Definition**. Given a proof π, define a totally rigid acyclic tree grammar $\mathrm{G}(\pi)$.
  (next slide: example)
- **Definition**. A proof is called *simple* if every cut-formula contains at most one quantifier.
- **Theorem**. Let π be a simple proof of $T \vdash \exists x\, A$ with A quantifier-free. Then $\mathrm{L}(\mathrm{G}(\pi)) = \{A[x \backslash t_1], \ldots, A[x \backslash t_n]\}$ and $T \vdash \bigvee_{i=1}^{n} A[x \backslash t_i]$ is provable.
  (in other words: $\mathrm{L}(\mathrm{G}(\pi))$ contains the witnesses for $\exists x\, A$)

Example

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{\vdash P(a), P(b)}{\vdash \exists x P(x), P(b)}\;\exists_r
}{\vdash \exists x P(x), \exists x P(x)}\;\exists_r
}{\vdash \exists x P(x)}\;c_r
\qquad
\dfrac{
\dfrac{
\dfrac{P(\alpha) \vdash Q(f(\alpha))}{P(\alpha) \vdash \exists x Q(x)}\;\exists_r
\qquad
\dfrac{
\dfrac{P(\alpha), Q(\beta) \vdash R(g(\alpha, \beta))}{P(\alpha), Q(\beta) \vdash \exists x R(x)}\;\exists_r
}{P(\alpha), \exists x Q(x) \vdash \exists x R(x)}\;\exists_l
}{P(\alpha) \vdash \exists x R(x)}\;c_l, \mathrm{cut}
}{\exists x P(x) \vdash \exists x R(x)}\;\exists_l
}{\vdash \exists x R(x)}\;\mathrm{cut}
$$

$\mathrm{G}(\pi) = \langle \varphi, R, \Sigma, P \rangle$ where $R = \{\varphi, \alpha, \beta\}$ and
$P = \{$

Example

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{\vdash P(a), P(b)}{\vdash \exists x P(x), P(b)}\,\exists_r
}{\vdash \exists x P(x), \exists x P(x)}\,\exists_r
}{\vdash \exists x P(x)}\,c_r
\qquad
\dfrac{
\dfrac{
\dfrac{P(\alpha) \vdash Q(f(\alpha))}{P(\alpha) \vdash \exists x Q(x)}\,\exists_r
\quad
\dfrac{
\dfrac{P(\alpha), Q(\beta) \vdash R(g(\alpha, \beta))}{P(\alpha), Q(\beta) \vdash \exists x R(x)}\,\exists_r
}{P(\alpha), \exists x Q(x) \vdash \exists x R(x)}\,\exists_l
}{P(\alpha) \vdash \exists x R(x)}\,c_l, \mathrm{cut}
}{\exists x P(x) \vdash \exists x R(x)}\,\exists_l
}{\vdash \exists x R(x)}\,\mathrm{cut}
$$

$G(\pi) = \langle \varphi, R, \Sigma, P \rangle$ where $R = \{\varphi, \alpha, \beta\}$ and
$P = \{\varphi \to R(g(\alpha, \beta))$

Example

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{\vdash P(a), P(b)}{\vdash \exists x P(x), P(b)}\;\exists_r
    }{\vdash \exists x P(x), \exists x P(x)}\;\exists_r
  }{\vdash \exists x P(x)}\;c_r
  \qquad
  \dfrac{
    \dfrac{
      \dfrac{P(\alpha) \vdash Q(f(\alpha))}{P(\alpha) \vdash \exists x Q(x)}\;\exists_r
      \quad
      \dfrac{
        \dfrac{P(\alpha), Q(\beta) \vdash R(g(\alpha, \beta))}{P(\alpha), Q(\beta) \vdash \exists x R(x)}\;\exists_r
      }{P(\alpha), \exists x Q(x) \vdash \exists x R(x)}\;\exists_l
    }{P(\alpha) \vdash \exists x R(x)}\;c_l, \text{cut}
  }{\exists x P(x) \vdash \exists x R(x)}\;\exists_l
}{\vdash \exists x R(x)}\;\text{cut}
$$

$\mathrm{G}(\pi) = \langle \varphi, R, \Sigma, P \rangle$ where $R = \{\varphi, \alpha, \beta\}$ and
$P = \{\varphi \to R(g(\alpha, \beta)),\ \beta \to f(\alpha)\}$

Example

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{\vdash P(a), P(b)}{\vdash \exists x P(x), P(b)}\ \exists_r
}{\vdash \exists x P(x), \exists x P(x)}\ \exists_r
}{\vdash \exists x P(x)}\ c_r
\qquad
\dfrac{
\dfrac{
\dfrac{P(\alpha) \vdash Q(f(\alpha))}{P(\alpha) \vdash \exists x Q(x)}\ \exists_r
\quad
\dfrac{
\dfrac{P(\alpha), Q(\beta) \vdash R(g(\alpha, \beta))}{P(\alpha), Q(\beta) \vdash \exists x R(x)}\ \exists_r
}{P(\alpha), \exists x Q(x) \vdash \exists x R(x)}\ \exists_l
}{P(\alpha) \vdash \exists x R(x)}\ c_l, \text{cut}
}{\exists x P(x) \vdash \exists x R(x)}\ \exists_l
}{\vdash \exists x R(x)}\ \text{cut}
$$

$\mathrm{G}(\pi) = \langle \varphi, R, \Sigma, P \rangle$ where $R = \{\varphi, \alpha, \beta\}$ and
$P = \{\varphi \to R(g(\alpha, \beta)),\ \beta \to f(\alpha), \alpha \to a, \alpha \to b\}$

Example

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{\vdash P(a), P(b)}{\vdash \exists x P(x), P(b)}\ \exists_r
    }{\vdash \exists x P(x), \exists x P(x)}\ \exists_r
  }{\vdash \exists x P(x)}\ c_r
  \qquad
  \dfrac{
    \dfrac{
      \dfrac{P(\alpha) \vdash Q(f(\alpha))}{P(\alpha) \vdash \exists x Q(x)}\ \exists_r
      \quad
      \dfrac{
        \dfrac{P(\alpha), Q(\beta) \vdash R(g(\alpha, \beta))}{P(\alpha), Q(\beta) \vdash \exists x R(x)}\ \exists_r
      }{P(\alpha), \exists x Q(x) \vdash \exists x R(x)}\ \exists_l
    }{P(\alpha) \vdash \exists x R(x)}\ c_l, \mathrm{cut}
  }{\exists x P(x) \vdash \exists x R(x)}\ \exists_l
}{\vdash \exists x R(x)}\ \mathrm{cut}
$$

$\mathrm{G}(\pi) = \langle \varphi, R, \Sigma, P \rangle$ where $R = \{\varphi, \alpha, \beta\}$ and
$P = \{\varphi \to R(g(\alpha, \beta)),\ \beta \to f(\alpha), \alpha \to a, \alpha \to b\}$

Hence $\mathrm{L}(\mathrm{G}(\pi)) = \{R(g(a, f(a))), R(g(b, f(b)))\}$

Outline

- √ Proof mining: cut-elimination
- √ Rigid tree languages
- √ From proofs to grammars
- ▶ From grammars to proofs

From Grammars to Proofs

- Given grammar find proof!
  Caveat: $\mathrm{L}(G)$ is a set of terms, $\mathrm{L}(\mathrm{G}(\pi))$ is a set of formulas
  $\Rightarrow$ Wrap up $\mathrm{L}(G)$ using a new predicate symbol
- **Theorem**. For every totally rigid acyclic tree grammar $G = \langle \beta, R, \Sigma, P \rangle$ there is a simple proof π with $\mathrm{G}(\pi) = \langle \alpha, R \cup \{\alpha\}, \Sigma, P \cup \{\alpha \to Q(\beta)\} \rangle$ s.t. cut-elimination of π computes $\mathrm{L}(\mathrm{G}(\pi))$.

$\Rightarrow$ Compression power of totally rigid acyclic tree grammars corresponds *exactly* to that of simple proofs.

$\Rightarrow$ Characterisation of class of proofs by class of grammars.

Conclusion

- Proofs and tree languages are intimately related

Applications / Future Work:

- Proof mining using tree grammars
- Cut-introduction (LPAR paper)
- Lower bounds on proofs
- Operations on languages get proof-theoretic meaning

Concrete Open Questions

- Go beyond simple proofs
- Does there exist a finite set T of terms s.t. every totally rigid acyclic tree grammar G with $\mathrm{L}(G) = T$ has $|G| = |T|$.
  (Uncompressible term-set $\Rightarrow$ lower bounds on proof length)
- What is the complexity of the problem: Given finite set T of terms, find minimal G with $\mathrm{L}(G) = T$?
  (Cut-introduction)
- Further cut-introduction algorithms