A TUTORIAL ON
PROOF THEORETIC FOUNDATIONS
OF
LOGIC PROGRAMMING

Paola Bruscoli & Alessio Guglielmi
Dresden Univ. of Technology
GERMANY

Mumbai, Dec. 2003
Motivations

- Logic programming ↔ resolution method

  ok with Horn clause language, but it is difficult to extend it keeping logical purity.

- Hint (Miller, Nadelhur, Henning, Scedrov '91)

  logic programming in proof theory
  (uniform provability)

  - declarative and operational meaning of logic programs coincide in several logics

  - for logic without a convenient semantics p.t. gives already a 'declarative' theoretical support [e.g. linear logic]

  - some properties of p.t. in these systems are understood as relevant properties of systems in automated deduction

- Essential proof theoretical tool

  Sequent Calculus (Gentzen)

  - cut-elimination property

  - → modularity in language design

- An existing language: λ-Prolog (int. h.o. logic)
  many more (hiu. logic)
1. Sequent Calculus & a system for classical logic (LK)
   - examples
   - cut-elimination (not proven)
   - use of cut-elimination
   - for the sake of readability, we move to intuitionistic logic (as syntactical restriction in LK)

2. An informal introduction to abstract logic programs
   - backchain: up in intuitionistic Lk
   - some desiderata:
     - provability in Lk is too much free!
   - uniform proofs (goal directed proof search)
     - abstract logic programming language (sample lemmata)

3. Abstract logic programming languages
   - system MNPS, intuitionistic
   - some examples
   - Horn clauses and Hereditary Harrop therein.
**Sequent Calculus (general)**

- **sequent**
  \[ \Gamma \vdash \Delta \]
  \[ \uparrow \uparrow \]
  multisets of formulae

  "from all formulae in \( \Gamma \), some of the formulae in \( \Delta \) follow."

- **rule**
  \[ g \quad S_1, \ldots, S_n \]
  \[ \frac{}{S} \]
  \[ \text{premises} \quad \text{conclusion} \]
  \[ \text{name} \]
  \[ (n=0 \Rightarrow \text{no premises: an axiom}) \]

- **System in the sequent calculus:** finite set of rules:
  - axiom
  - left rules
  - right rules
  (axiom | structural (L/R) | left rules | right rules)
  + cut
- **Structural rules:** model basic algebraic properties of sequents

- **derivation for \( \Delta \):** apply successively the given rules. If all branches of the deriv. reach the axioms, then it is a proof. (bottom-up perspective)
SYSTEM LK (first order classical logic, multi-set cut)

[Geuken '34]

Axion

\[ \Gamma \vdash A \]

Cut

\[ \frac{\Gamma, \Delta, A \quad A, \Delta \vdash \Theta}{\Gamma, \Delta, A, \Theta} \]

Structural Rules

**Left Rules**

- **CL**
  \[ A, A, \Gamma \vdash \Delta \implies A, \Gamma \vdash \Delta \]

- **CR**
  \[ \Gamma, \Delta, A, A \implies \Gamma, \Delta, A \]

- **WL**
  \[ A, \Gamma \vdash A \]

- **WR**
  \[ \Gamma \vdash \Delta, A \]

**Right Rules**

- **DL**
  \[ \Gamma, \Delta, A, B, \Gamma \vdash \Delta \implies A \land B, \Gamma \vdash \Delta \]

- **DR**
  \[ A, \Gamma \vdash \Delta, B \implies \Gamma \vdash \Delta, A \lor B \]

- **VR**
  \[ A, \Gamma \vdash \Delta, B \implies A \lor B, \Gamma \vdash \Delta \]

- **VR**
  \[ \Gamma \vdash \Delta, A \land B \]

- **VL**
  \[ A, \Gamma \vdash \Delta, B, \Gamma \vdash \Delta \implies A \lor B, \Gamma \vdash \Delta \]

- **VR**
  \[ A, \Gamma \vdash \Delta, B \implies A \lor B, \Gamma \vdash \Delta \]

where in **\( \forall \)** and **\( \exists \)** the variable \( x \) is not free in the conclusion

\[ \Gamma, \Delta \text{ multisets} \]

[bound variables can always be renamed: \( \forall x. p(x) \) is the same as \( \forall y. p(y) \)]
ADMISSIBILITY OF CUT

* Cut: the only rule which doesn't meet the subformula property

* Bad for automatic deduction

* Useful when dealing with semantics or when relating different systems for the same logic, possibly given in different formalism (n.d.)

* Hence, systems in sequent calculus usually have a cut

Key point: is the cut disposable?

"admissibility": the other rules are sufficient for the completeness of the system

Gentzen's Thm:
For every proof in LK there is a proof with the same conclusion which doesn't use the cut

- constructive proof
- relevant for functional programming
There is a procedure to transform proofs with cut into cut-free proofs (see e.g. Gallier).

The cut-free proof is usually bigger than the one with cut (hyperexponential factor in the size).

Less intuitive from the semantical point of view.

Cut-free proof for: \( \exists x. \forall y. (p(x) \supset p(y)) \)

\[
\frac{\vdash p(y) \supset p(y) }{\text{W}L} \\
\frac{p(y), p(x) \vdash p(y) }{\text{C}R} \\
\frac{p(y) \vdash p(x) \supset p(y) }{\text{W}R} \\
\frac{p(y) \vdash p(z), p(x) \supset p(y) }{\text{C}R} \\
\frac{\vdash p(y) \supset p(z), p(x) \supset p(y) }{\text{A}R} \\
\frac{\vdash \forall z. (p(y) \supset p(z)), p(x) \supset p(y) }{\text{E}R} \\
\frac{\vdash \exists x. \forall y. (p(x) \supset p(y)), p(x) \supset p(y) }{\text{A}R} \\
\frac{\vdash \exists x. \forall y. (p(x) \supset p(y)), \forall y. (p(x) \supset p(y)) }{\text{E}R} \\
\frac{\vdash \exists x. \forall y. (p(x) \supset p(y)), \exists y. (p(x) \supset p(y)) }{\text{E}R} \\
\frac{\vdash \exists x. \forall y. (p(x) \supset p(y)) }{\text{C}R}
\]
Prove that LK is consistent:

By contradiction:

- LK inconsistent $\Rightarrow$ $\exists \ \Pi_1 \ 1 \ \Pi_2 \ 1 \ \neg A$

- Then the following proof for $1$ exist:

  $\Pi_1 \ 1 \ A$

  $\Pi_2 \ 1 \ \neg A$

  cut $1 \ A$

- By the cut-elim. thm. there must be a cut-free proof for $1$

- Impossible because
  1) $1$ is not an axiom
  2) $1$ is not the conclusion of any rule in LK other than cut
Every axiom $A+A$ in a proof can be replaced by using axioms of the kind $a+a$ (a atom, $A$ formula).

By induction on the nr. of logical connectives in $A$:

A atom: $\pi_A = A+A$
Logic programming problem

Find a substitution $\sigma$ s.t. in some formal system the following is provable:

$$\text{P} \vdash (a_1 \land \ldots \land a_k) \sigma$$

(collection of clauses)

We take:

- provability in $\text{LK}$
- $\text{P}$ multiset of clauses

INTUITIONISTIC LOGIC

(sequents with at least one formula on the right)
\[
\forall x_1 \ldots \forall x_n. (b_1 x_1 \ldots b_n x_n \supset a) \quad \text{in } P
\]

\[
(a_1 \land \ldots \land a_k) \quad \text{goal}
\]

\(\Delta:\)

\[
\begin{array}{c}
\vdash \neg \neg a_1 \quad \ldots \\
\vdash \neg \neg a_k
\end{array}
\]

\[
\vdash a_1 \land \ldots \land a_k
\]

(similar for \(h=0\))

**BACKCHAIN RULE**

\[
\begin{array}{c}
\vdash a_1 \circ \quad \vdash a_k \circ \\
\vdash \neg \neg (b_1 x_1 \ldots b_n x_n) \circ \\
\vdash a_2 \circ \quad \vdash a_k \circ
\end{array}
\]

\[
\vdash a_1 \land \ldots \land a_k \circ
\]

\(b\) degenerates to an axiom when \(h=0\), \(k=1\).
\[ \forall x. ((b_1 \land \ldots \land b_k) \Rightarrow a) \quad \text{in } P \]
\[ (a_1 \land \ldots \land a_k) \quad \text{goal} \]
\[ \circ \text{ is unifier of } a, a_e \text{ for some } 1 \leq e \leq k \]

\[ \Delta: \]
\[ (h \circ 0) \]

\[ \forall_l \]
\[ \vdash P, \forall x. ((b_1 \land \ldots \land b_k) \Rightarrow a) \Rightarrow a_e \]
\[ \vdash a_e \]
\[ \vdash a_k \]

\[ \vdash (a_1 \land \ldots \land a_k) \]

(similar for h=0)

**Backchain Rule**

\[ \vdash a_1 \circ \]
\[ \vdash a_2 \circ \]
\[ \vdash (b_1 \land \ldots \land b_k) \circ \]
\[ \vdash a_{e+1} \circ \]
\[ \vdash a_k \circ \]

\[ \vdash (a_1 \land \ldots \land a_k) \circ \]

(b degenerates to an axiom when h=0, k=1.)
**Abstract Logic Programming:** backchain

- \( \forall x. ((b_1 \land \ldots \land b_n) \Rightarrow a) \) in \( P \)
- \( (a_1 \land \ldots \land a_k) \) goal
- \( \sigma \) is unifies of \( a, a_e \) for some \( 1 \leq e \leq k \)

\[
\Delta
\]

\[ (h \neq 0) \]

\[
\frac{\Pr(b_1 \land \ldots \land b_n) \sigma}{P, a \sigma \Rightarrow a_e \sigma}
\]

\[
\frac{\Pr(b_1 \land \ldots \land b_n) \sigma, a_e \sigma}{P, a \sigma \Rightarrow a_e \sigma}
\]

\[
\frac{P, (b_1 \land \ldots \land b_n) \Rightarrow a \sigma \Rightarrow a_e \sigma}{P, a \sigma \Rightarrow a_e \sigma}
\]

\[
\frac{\Pr a_1 \sigma}{P, a \sigma \Rightarrow a_e \sigma}
\]

\[
\frac{\Pr a_k \sigma}{P, a \sigma \Rightarrow a_e \sigma}
\]

\[
\Pr(a_1 \land \ldots \land a_k) \sigma
\]

(similar for \( n = 0 \))

**Backchain Rule**

\[
\Pr a_1 \sigma \quad \Pr a_{k-1} \sigma \quad \Pr(b_1 \land \ldots \land b_n) \sigma \quad \Pr a_{k+1} \sigma \quad \Pr a_k \sigma
\]

\[
\Pr(a_1 \land \ldots \land a_k) \sigma
\]

\( b \) degenerates to an axiom when \( h = 0 \) \( k = 1 \).
Abstract Logic Programming: backchain

- (bottom-up reading) it reduces one logic programming problem to zero or more logic programming problems

- To every application of \( b \) it is associated a \( \sigma \) which allows us to use an axiom leaf

- \( \sigma \) must not be necessarily the main!

- We call \( \sigma \) the answer substitution of that application of \( b \)

- The composition of all \( \sigma \) is the answer substitution to the original logic programming problem.

→ look for proofs of logic programming problems, built only by applying \( b \)

Proof SEARCH SPACE:

- root: l.p. problem
- model: derivations \( (b) \)
- father/child: apply one \( b \) more
  (cautious to substitution)
\( (x)d \vdash q \)

\[
\frac{(x)d \vdash q}{(x)\bot \lor (x)b \vdash q}
\]

\[
\frac{(x)d \vdash q}{(x)b \vdash q}
\]

\[
\frac{(x)b \vdash q}{(x)b \bot \vdash q}
\]

\[
\frac{(x)b \bot \vdash q}{(x)b \vdash q}
\]

\[
\frac{(x)b \vdash q}{(x)\bot \lor (x)b \vdash q}
\]

\[
\frac{(x)\bot \lor (x)b \vdash q}{(x)d \vdash q}
\]

\[
\frac{(x)d \vdash q}{(x)b \bot \vdash q}
\]
\[ P \vdash a \]

\[ P = \{ a \land b \Rightarrow a \} \]
- no def. of backchain, yet but
  it transforms our logic programming problem into zero
  of more

- let the goal drive the computation:
  goal is a set of instructions to the program

Ex:

Prove $A \land B$ in the program $P$:
- prove $A$ in $P$
- prove $B$ in $P$

\[
\frac{P \vdash A \quad P \vdash B}{P \vdash A \land B}
\]

(intuitionistic logic)

... but in Lt there are many more options, ex. use left rules.

(Ritter, Nadezhda Pfenning, Scedro)

1. Define what strategy for searching for proofs could correspond to a sensible operational semantics (UNIFORM PROOFS)

2. When does UNIFORM PROVABILITY correspond to (the usual, unrestricted) PROVABILITY? (ABSTRACT LOGIC PROGRAMMING LANGUAGES)
**Axioms**

\[ a, \Gamma \vdash \Delta, \alpha \]

\[ \bot, \Gamma \vdash \Delta, \bot \]

\[ \Gamma \vdash \Delta, \top \]

**Left Rules**

\[ \forall L \quad \frac{\Gamma \vdash \Delta, A, B, \Gamma \vdash \Theta}{A \land B, \Gamma \vdash \Delta, \Theta} \]

\[ \forall L \quad \frac{A, B, \Gamma \vdash \Delta}{A \land B, \Gamma \vdash \Delta} \]

\[ \forall L \quad \frac{A, \Gamma \vdash \Delta, B, \Gamma \vdash \Delta}{A \lor B, \Gamma \vdash \Delta} \]

\[ \forall L \quad \frac{A[t/x], \Gamma \vdash \Delta}{\forall x. A, \Gamma \vdash \Delta} \]

\[ \exists L \quad \frac{A, \Gamma \vdash \Delta}{\exists x. A, \Gamma \vdash \Delta} \]

**Right Rules**

\[ \exists R \quad \frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \lor B} \]

\[ \forall R \quad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, \forall x. A} \]

\[ \forall R \quad \frac{\Gamma \vdash \Delta, A[t/x]}{\Gamma \vdash \Delta, \exists x. A} \]

\[ \forall R \quad \frac{\Gamma \vdash \Delta, \bot}{\Gamma \vdash \Delta, \alpha} \text{ where } A \neq \bot \]

*where in \[ \forall R \text{ and } \exists L \] the variable \( x \) is not free in the conclusion*

- **Cut Free!**
- **no structural rules** → their 'functionality' is in the shape of sequents
- \( \Gamma \vdash \Delta \quad \Gamma \vdash \Delta \text{ sets} \)
  - \( \top, \bot \) (not atoms)
- **negation:** \[ \neg A \equiv A \equiv \bot \]

**MNPS is equivalent to LK**

→ Consider its intutionistic version: at most one formula at the right
I-proof: a proof in MNPS where each sequent has a singleton set as its r.h.s.

Uniform proof: an I-proof in which each occurrence of a sequent, whose r.h.s. contains a non-atomic formula is the lower sequent of the inference rule that introduces its top-level connective.

[Reduce the goal to atomic form by applying right rules, and only then start applying left rules.]

Ex

Find a uniform proof for

\[(\land b) \supset ((\land b) \supset 1) \supset 1\]

Ex

Backchain rule obeys the def.

But

Uniform provability, for classical and intuitionistic logic, does not coincide with provability.

1. some stunts are provable, but not uniformly provable
2. depending on the instance of the stunt, there might be or not a uniform proof.

So, the language matters too!
1. $\forall x . p(x) \Rightarrow \exists x . \neg p(x)$

$\square_L$

$\forall x . p(x), p(t) \Rightarrow \bot$

$\square_R$

$\neg \forall x . p(x) \Rightarrow \neg p(t)$

$\exists_R$

$\neg \forall x . p(x) \Rightarrow \exists x . \neg p(x)$

$\exists_R$

$\neg \forall x . p(x) \Rightarrow \exists x . \neg p(x)$

2. $(a \lor b) \Rightarrow (b \lor a)$

$\forall_i$

$a \lor b \Rightarrow b \lor b$

$\forall_k$

$a \lor b \Rightarrow b \lor a$

$\exists_k$

$(a \lor b) \Rightarrow (b \lor a)$

3. $A \Rightarrow ((A \Rightarrow \bot) \Rightarrow \bot)$

$[A \Rightarrow \neg (A)]$

ok if $A = a \lor b$, but for example, no way to build a uniform proof for

$A = \forall x . (p(x) \lor q(x))$
ABSTRACT LOGIC PROGRAMMING LANGUAGES

Aim
Define a sort of completeness notion, to say that for a particular language, provability and uniform provability coincide.

We want to keep the idea of uniform provability.

\[ \langle D, G, \vdash \rangle \]
\[ P, G \]

clauses and goals

\[ P = \text{finite set of clauses} \]
\[ GE\ G \]

\[ \langle D, G, \vdash \rangle \] abstract logic programming language if whenever \( P \vdash G \) provable, then \( P \vdash G \) is uniformly provable, (\( \forall P, VG. \))

how to prove if a language is an a.l.p.l.
study the permutable relations among inference rules of the system defining \( \vdash \).

Ex.

\[ \frac{B, \Gamma \vdash C}{\Gamma \vdash A, B, \Gamma \vdash C} \]
\[ \frac{\Gamma \vdash A \quad B, \Gamma \vdash C}{A \vdash B, \Gamma \vdash C} \]

(not uniform)

\[ \frac{\Gamma \vdash A \quad B, \Gamma \vdash C}{A \vdash B, \Gamma \vdash CVD} \]

(uniform)
Examples of A.L.T.L.

\[ G := A \mid D \supset A \mid G \lor G \]
\[ D := A \mid G \supset A \mid \forall x. D \]
\[ \vdash : \text{NNPS (intuitionistic)} \]

- We can define a backchain rule:

\[
\frac{\quad b \quad}{P, D \supset G, \sigma} \quad \frac{P, (G \supset \alpha'), \sigma, D \supset a}{P, (G \supset \alpha'), D \supset a}
\]

\[
\frac{\quad b \quad}{P, D \supset G, \sigma} \quad \frac{P, (G \supset \alpha'), \sigma, D \supset a}{P, (G \supset \alpha'), D \supset a}
\]

\[
\frac{\quad b \quad}{P, D \supset G, \sigma} \quad \frac{P, (G \supset \alpha'), \sigma, D \supset a}{P, (G \supset \alpha'), D \supset a}
\]

\[
\frac{\quad b \quad}{P, D \supset G, \sigma} \quad \frac{P, (G \supset \alpha'), \sigma, D \supset a}{P, (G \supset \alpha'), D \supset a}
\]

\[
\frac{\quad b \quad}{P, D \supset G, \sigma} \quad \frac{P, (G \supset \alpha'), \sigma, D \supset a}{P, (G \supset \alpha'), D \supset a}
\]

\[
\frac{\quad b \quad}{P, D \supset G, \sigma} \quad \frac{P, (G \supset \alpha'), \sigma, D \supset a}{P, (G \supset \alpha'), D \supset a}
\]

\[
\frac{\quad b \quad}{P, D \supset G, \sigma} \quad \frac{P, (G \supset \alpha'), \sigma, D \supset a}{P, (G \supset \alpha'), D \supset a}
\]

\[
\frac{\quad b \quad}{P, D \supset G, \sigma} \quad \frac{P, (G \supset \alpha'), \sigma, D \supset a}{P, (G \supset \alpha'), D \supset a}
\]
\[ \text{Do, Da, D2} \vdash \text{Ga, G2} \]

\[ \text{Do} \Rightarrow (\forall x. (D_{a} \Rightarrow G_{a}) \land \forall y. (D_{2} \Rightarrow G_{2})) \]

\[ \frac{\text{Do, Da} \vdash G_{a}}{	ext{Dr}} \frac{\text{Do, D2} \vdash G_{2}}{	ext{Dr}} \]

\[ \frac{\frac{\text{Do} \Rightarrow \forall x. (D_{a} \Rightarrow G_{a})}{\text{Dr}}}{\text{Dr}} \frac{\frac{\text{Do} \Rightarrow \forall y. (D_{2} \Rightarrow G_{2})}{\text{Dr}}}{\text{Dr}} \]

\[ \frac{\text{Do} \Rightarrow (\forall x. (D_{a} \Rightarrow G_{a}) \land \forall y. (D_{2} \Rightarrow G_{2}))}{\text{Dr}} \]
HORN CLAUSES and HEREDITARY HARROP FORMULAE

- both are proven to be a.l.p.l. (Miller, Nederpelt, Pfenning, Sedun)

- Horn clauses

\[ G ::= T | A | G \lor G | G \land G | \exists x . G \]
\[ D ::= A | G \Rightarrow D \lor D \lor \forall x . D \]

Limitation of Horn clauses: inability to deal with modules
\rightarrow move to

- Hereditary Harrop Formulae:

\[ G ::= T | A | G \lor G | G \land G | \exists x . G | \forall x . G | D \lor G \]
\[ D ::= A | G \Rightarrow D \lor D \lor \forall x . D \]

Implication in goals \rightarrow module
Universal quantification in goals \rightarrow private variables in modules

- Higher order

quantify over variables and relations \rightarrow abstract data types
There is much more, related to more exotic logics, or new logics.

The proof theoretical foundation is especially useful when the semantics are too complicated or, simply, not available.

Related to the design of logic programming languages:

- **proof theoretical properties**
  - (cut elimination, atomicity of cut/axiom, ...)

  ▲

- properties required by *system designers*
  - (modularity, locality)

→ Beyond the sequent calculus: the calculus of structures
CALCULUS OF STRUCTURES

A. Gagglui + K. Brünnler
P. Bruscol
C. Stewart
P. Stroupa
L. Straßburger
O. Kahramanogullari
A. Tiu
R. Hein

- Any critic to the sequent calculus?
  - drop the idea of main connective
  - formulae and sequents merged into a structure
  - deep (contextual) rewriting
  - top-down symmetries

- A completely new proof theory
  modularity, locality, cut-elimination, interpolation,
  decomposition
  classical, linear (MLL, MELL, MALL), linear + seq,
  modal, (intuitionistic)

- Systems in the S.C. benefit from a presentation in CcS
- There exists a system that cannot be presented in S.C
  (system \(BV = MLL + \text{mix} + \text{seq}\))
  counterexample by Tiu

- UNIFORM PROVABILITY?