

JProver: Integrating Connection-based Theorem Proving into Interactive Proof Assistants

Stephan Schmitt¹, Lori Lorigo², Christoph Kreitz², Aleksey Nogin²

¹Dept. of Sciences and Engineering
Saint Louis University (Madrid Campus)
Madrid, Spain



²Dept. of Computer Science
Cornell University
Ithaca, NY 14853



MOTIVATION

● Interactive Proof Assistants

- Large scale applications of automated reasoning
- Expressive logics vs. higher degree of automation
- Coq, HOL, Isabelle, Nuprl, OMEGA, PVS

● Improving Proof Automation

- Proof planning for induction / first-order logic (HOL+CLAM / OMEGA+OTTER)
- Decision procedures, e.g. for fragments of arithmetic (HOL, Nuprl, STeP)
- Automatic theorem provers for first-order logics (HOL, Nuprl)

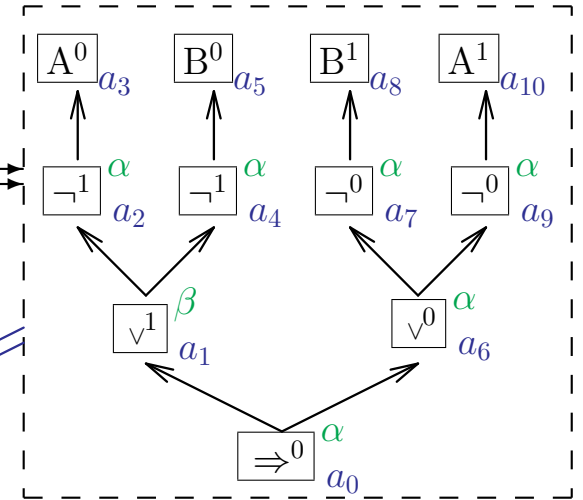
● JProver: Constructive logics

- Complete theorem prover for first-order intuitionistic logic
- Modular interface for connecting to interactive proof assistants
- Integrated into Nuprl / MetaPRL

THE AUTOMATED THEOREM PROVER

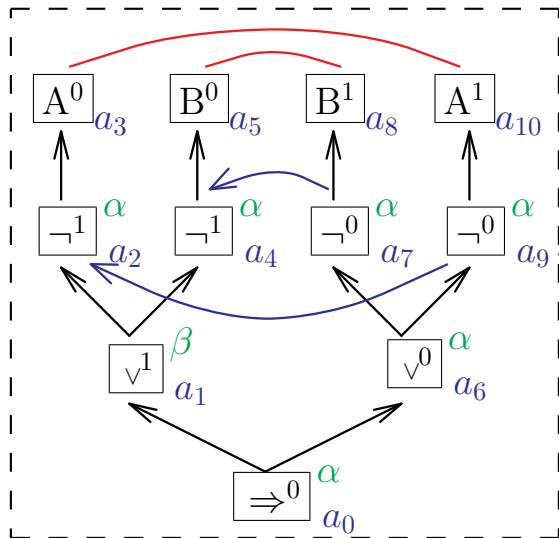
Formula

$$\neg A \vee \neg B \Rightarrow \neg B \vee \neg A$$



Matrix prover

= connection-driven path checking
 + intuitionistic string unification
 Substitutions induce ordering
 Otten & Kreitz '96, Kreitz & Otten '99



Proof Transformation

Search-free traversal of \triangleleft
 multiple \rightarrow single-conclusion
 Kreitz & Schmitt'00, Schmitt'00,
 Egly & Schmitt'99

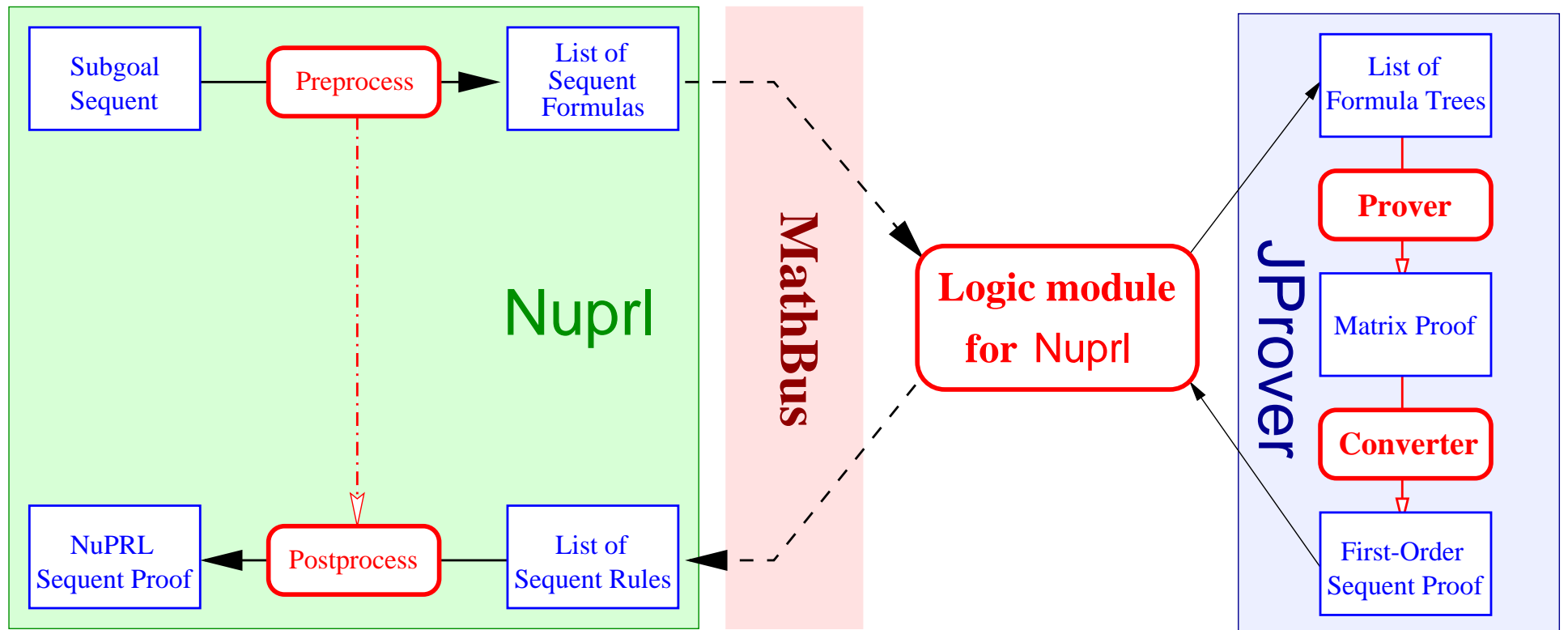
$$\frac{\frac{\frac{A \vdash A}{\neg A, A \vdash} \neg l \quad ax.}{\neg A \vdash \neg B, \neg A} \neg r \quad \frac{\frac{B \vdash B}{\neg B, B \vdash} \neg l \quad ax.}{\neg B \vdash \neg B, \neg A} \neg r}{\frac{\neg A \vee \neg B \vdash \neg B, \neg A}{\neg A \vee \neg B \vdash \neg B \vee \neg A} \vee r} \Rightarrow r \quad \frac{}{\vdash \neg A \vee \neg B \Rightarrow \neg B \vee \neg A} \Rightarrow r$$

Reduction Ordering \triangleleft

Sequent Proof

s

JProver INTEGRATION ARCHITECTURE



INTEGRATION INTO PROOF ASSISTANTS

● Logic Module: Required Components

- OCaml code communicating with proof assistant
- JLogic module representing the proof assistant's logic

● The JLogic module

- Describes terms implementing logical connectives
- Provides operations to access subterms
- Decodes sequent received from communication code
- Encodes JProver's sequent proof into format for communication code

```
module Nuprl_JLogic =
struct
let is_all_term = nuprl_is_all_term
let dest_all = nuprl_dest_all
let is_exists_term = nuprl_is_exists_term
let dest_exists = nuprl_dest_exists
let is_and_term = nuprl_is_and_term
let dest_and = nuprl_dest_and
let is_or_term = nuprl_is_or_term
let dest_or = nuprl_dest_or
let is_implies_term = nuprl_is_implies_term
let dest_implies = nuprl_dest_implies
let is_not_term = nuprl_is_not_term
let dest_not = nuprl_dest_not
type inference = '(string*term*term) list
let empty_inf = []
let append_inf inf t1 t2 r =
  ((Jall.ruleable r), t1, t2) :: inf
end
```

INTEGRATION INTO Nuprl / MetaPRL

● Connection to MetaPRL:

- JProver is a module in MetaPRL's code base
- MetaPRL communicates with JProver making a function call
- MetaPRL formulas are passed directly to JProver
- JLogic module converts sequent proof into MetaPRL tactic

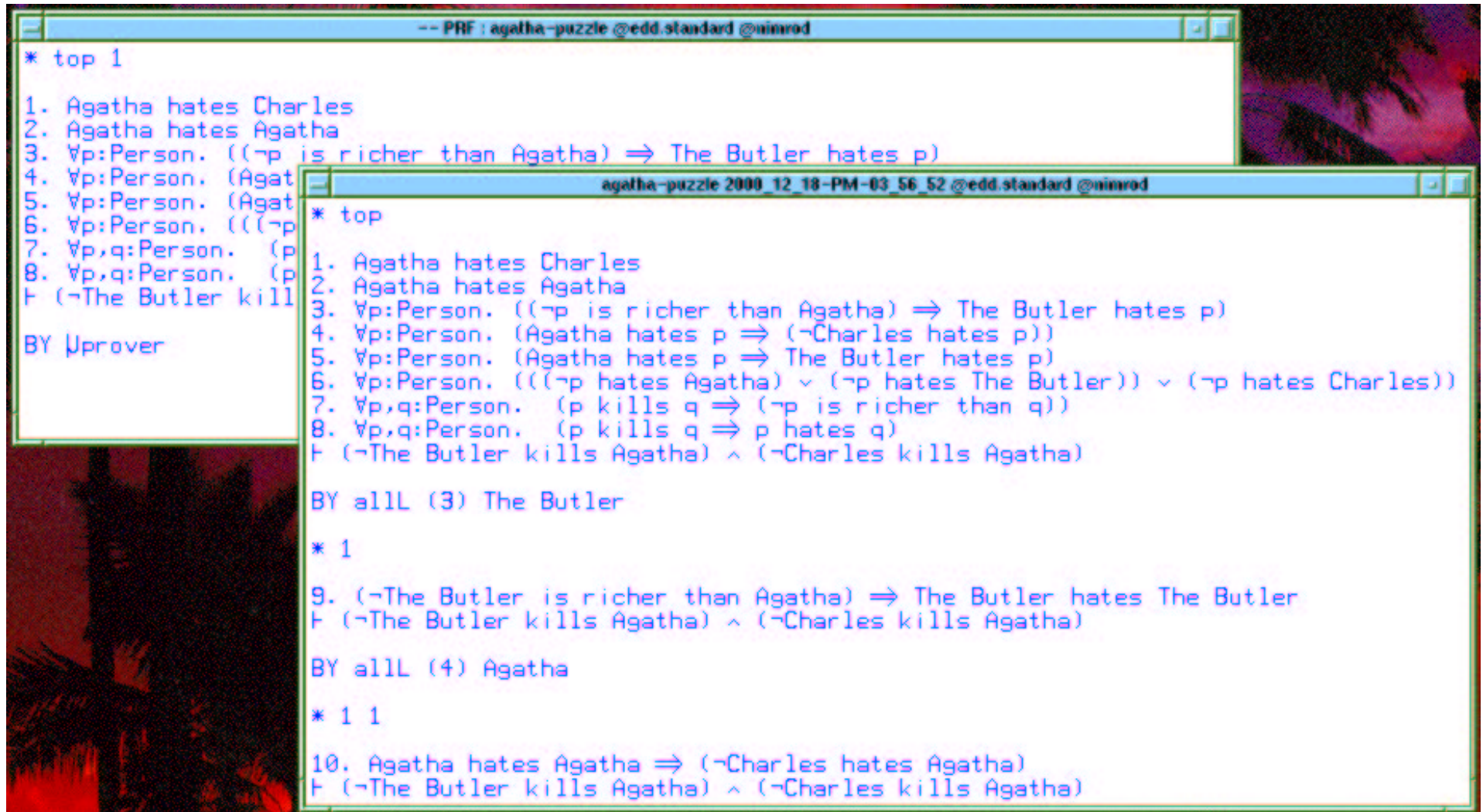
● Connection to Nuprl

- Preprocesses Nuprl sequent and semantical differences
- Sends terms in MathBus format over an INET socket
- JLogic module accesses semantical information from terms; converts sequent proof into format Nuprl can interpret
- Postprocesses result into Nuprl proof tree for original sequent

● Proof Validation

- Nuprl and MetaPRL do not rely on correctness of JProver
- JProver's output executed on original sequents in the systems

EXAMPLE: THE “AGATHA MURDER PUZZLE”



```
-- PRF : agatha-puzzle @edd.standard @ninrod
* top 1
1. Agatha hates Charles
2. Agatha hates Agatha
3.  $\forall p:\text{Person. } ((\neg p \text{ is richer than Agatha}) \Rightarrow \text{The Butler hates } p)$ 
4.  $\forall p:\text{Person. } (\text{Agatha hates } p \Rightarrow (\neg \text{Charles hates } p))$ 
5.  $\forall p:\text{Person. } (\text{Agatha hates } p \Rightarrow \text{The Butler hates } p)$ 
6.  $\forall p:\text{Person. } (((\neg p \text{ hates Agatha}) \vee (\neg p \text{ hates The Butler})) \vee (\neg p \text{ hates Charles}))$ 
7.  $\forall p,q:\text{Person. } (p \text{ kills } q \Rightarrow (\neg p \text{ is richer than } q))$ 
8.  $\forall p,q:\text{Person. } (p \text{ kills } q \Rightarrow p \text{ hates } q)$ 
 $\vdash (\neg \text{The Butler kills Agatha}) \wedge (\neg \text{Charles kills Agatha})$ 
BY  $\uparrow$ prover

agatha-puzzle 2000_12_18-PM-03_56_52 @edd.standard @ninrod
* top
1. Agatha hates Charles
2. Agatha hates Agatha
3.  $\forall p:\text{Person. } ((\neg p \text{ is richer than Agatha}) \Rightarrow \text{The Butler hates } p)$ 
4.  $\forall p:\text{Person. } (\text{Agatha hates } p \Rightarrow (\neg \text{Charles hates } p))$ 
5.  $\forall p:\text{Person. } (\text{Agatha hates } p \Rightarrow \text{The Butler hates } p)$ 
6.  $\forall p:\text{Person. } (((\neg p \text{ hates Agatha}) \vee (\neg p \text{ hates The Butler})) \vee (\neg p \text{ hates Charles}))$ 
7.  $\forall p,q:\text{Person. } (p \text{ kills } q \Rightarrow (\neg p \text{ is richer than } q))$ 
8.  $\forall p,q:\text{Person. } (p \text{ kills } q \Rightarrow p \text{ hates } q)$ 
 $\vdash (\neg \text{The Butler kills Agatha}) \wedge (\neg \text{Charles kills Agatha})$ 
BY allL (3) The Butler
* 1
9.  $(\neg \text{The Butler is richer than Agatha}) \Rightarrow \text{The Butler hates The Butler}$ 
 $\vdash (\neg \text{The Butler kills Agatha}) \wedge (\neg \text{Charles kills Agatha})$ 
BY allL (4) Agatha
* 1 1
10.  $\text{Agatha hates Agatha} \Rightarrow (\neg \text{Charles hates Agatha})$ 
 $\vdash (\neg \text{The Butler kills Agatha}) \wedge (\neg \text{Charles kills Agatha})$ 
```

CONCLUSION

● Progress

- Hybrid proofs: multiple provers with different formalisms
 - = expressive power of proof assistants for complex proofs / verifications
 - + efficient proof techniques for first-order subproblems
- Dealing with type information: discard or encode as predicates
- JProver applicable to proof problems beyond first-order logic

● Future Work

- Improve JProver's performance
- Combine JProver with Nuprl tactics and decision procedures
- Extend JProver to modal logics and inductive theorem proving
(Kreitz & Otten 1999, Kreitz & Pientka 2001)

● Demonstration

- Calling JProver from Nuprl: proof examples