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

Stephan Schmitt[1], Lori Lorigo[2], Christoph Kreitz[2], Aleksey Nogin[2]

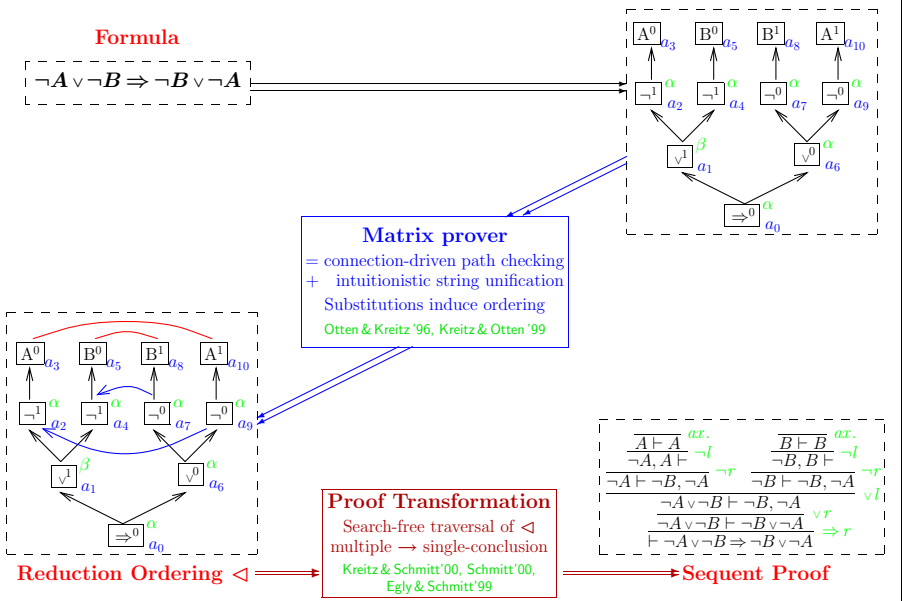
[1]Dept. of Sciences and Engineering
Saint Louis University (Madrid Campus)
Madrid, Spain

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

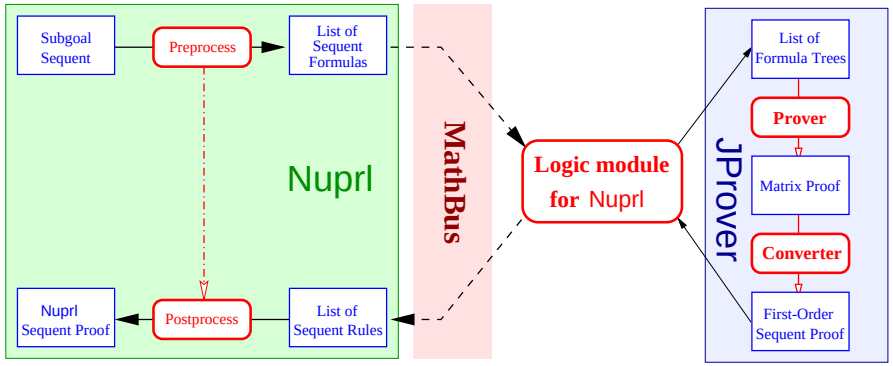
Motivation

- **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



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.ruletable r), t1, t2) :: inf
end
```

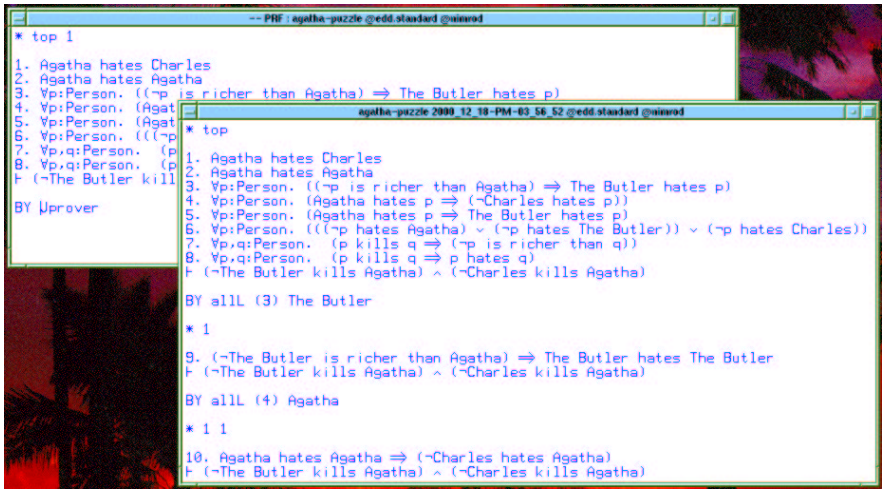

Integration into Nuprl / MetaPRL

- **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"



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