How to Prove Inductive Theorems?

QUODLIBET!

(= As you like it!)

Jürgen Avenhaus	Univ. Kaiserslautern	Germany
Ulrich Kühler	sd&m AG, Ratingen	Germany
Tobias Schmidt-Samoa	Univ. Kaiserslautern	Germany
Claus-Peter Wirth	Saarland Univ.	Germany

Why another Inductive Theorem Prover?

- Overcome limitations of
  Explicit Induction Paradigm (EIP)

Weaker Admissibility Conditions than EIP

Specification with partial + non-terminating function definitions incl. mutual + destructor recursion

$$
\begin{array}{rcll}
x - 0 & = & x & \\
\mathsf{s}(x) - \mathsf{s}(y) & = & x - y & \\
x \,/\, y & = & 0 & \text{if} \quad y \neq 0, \;\; x < y \\
x \,/\, y & = & \mathsf{s}((x - y) \,/\, y) & \text{if} \quad y \neq 0, \;\; x \not< y
\end{array}
$$

$$
\begin{array}{rcll}
0 - \mathsf{s}(y) & = & 666 & \\
x \,/\, y & = & 999 & \text{if} \quad y = 0
\end{array}
$$

Semantics?

Recursion analysis?

Proving?

Why another Inductive Theorem Prover?

- Overcome limitations of
  Explicit Induction Paradigm (EIP)
- Towards a Working Mathematician's style
  (*Descente Infinie*)

Descente Infinie?

In Mathematics:

- Used from Hippasos to Euclid
- Fermat first described and named the method
- Used to search for all hard induction proofs ever since

In ATP:

(Martin Protzen, CADE 1994)

Lazy Generation of Induction Hypotheses.

To overcome limitations of recursion analysis

Working Mathematician's Style

1. Simplify the conjecture in case analysis.
2. When appropriate:
   Apply conjecture just like a lemma
   (actually: application as an *induction hypothesis*).
3. Search for a single wellfounded ordering in which all induction hypotheses are smaller than the conclusion.

QUODLIBET **provides the Flexibility...**

... needed for searching for hard induction proofs:

- Generation of induction hypotheses:
  eager / lazy / mutual
- Choice of induction Ordering: eager / lazy
- Open lemmas
- Alternative proof attempts in parallel:
  forest of and/or-trees

Why another Inductive Theorem Prover?

- Overcome limitations of
  Explicit Induction Paradigm (EIP)
- Towards a Working Mathematician's style
  (*Descente Infinie*)
- Graceful degradation when automation fails

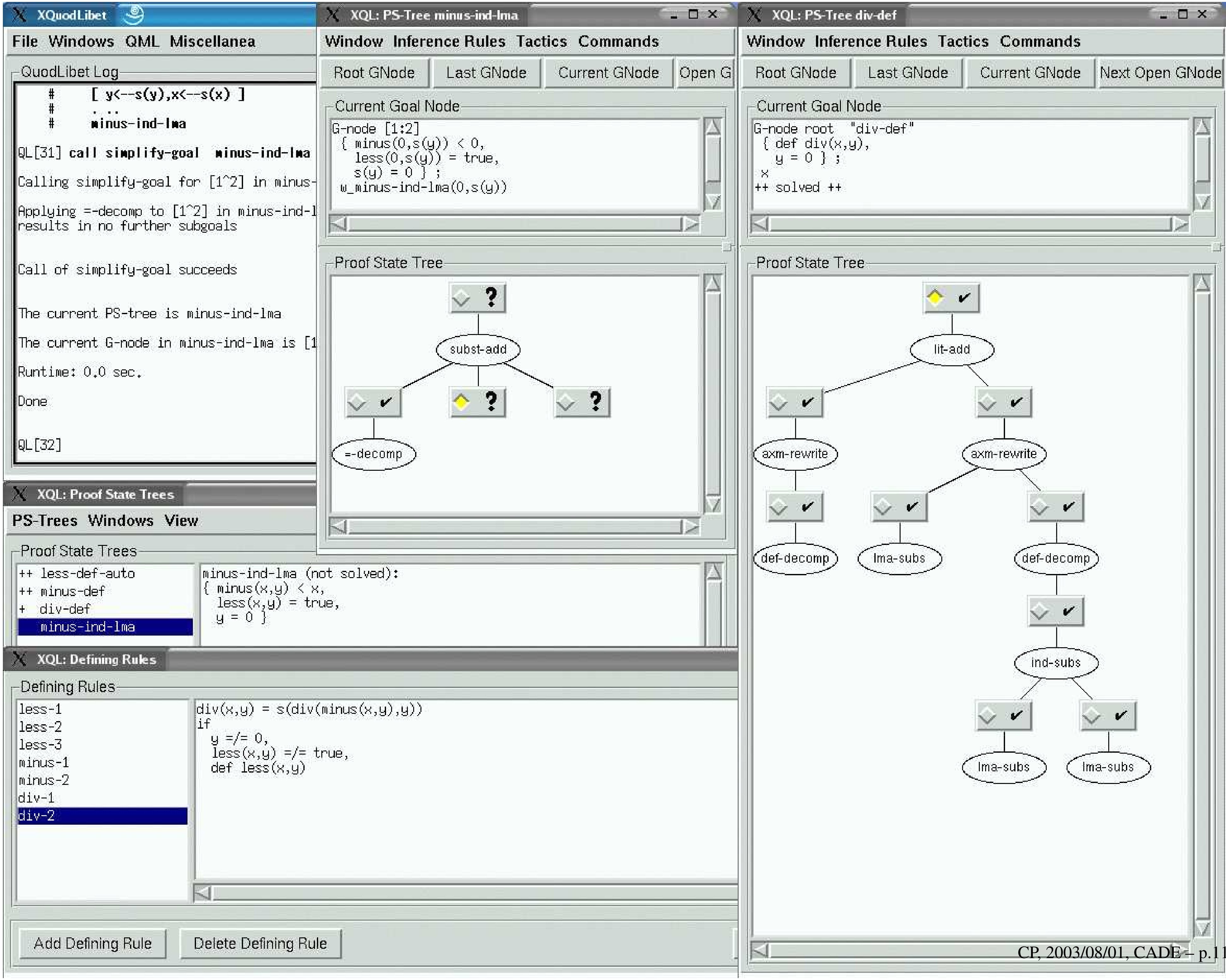
Graceful degradation

If tactics fail, they realize this early and ask the user for advice by presenting the current proof state as a tree in the GUI, which contains all relevant information.

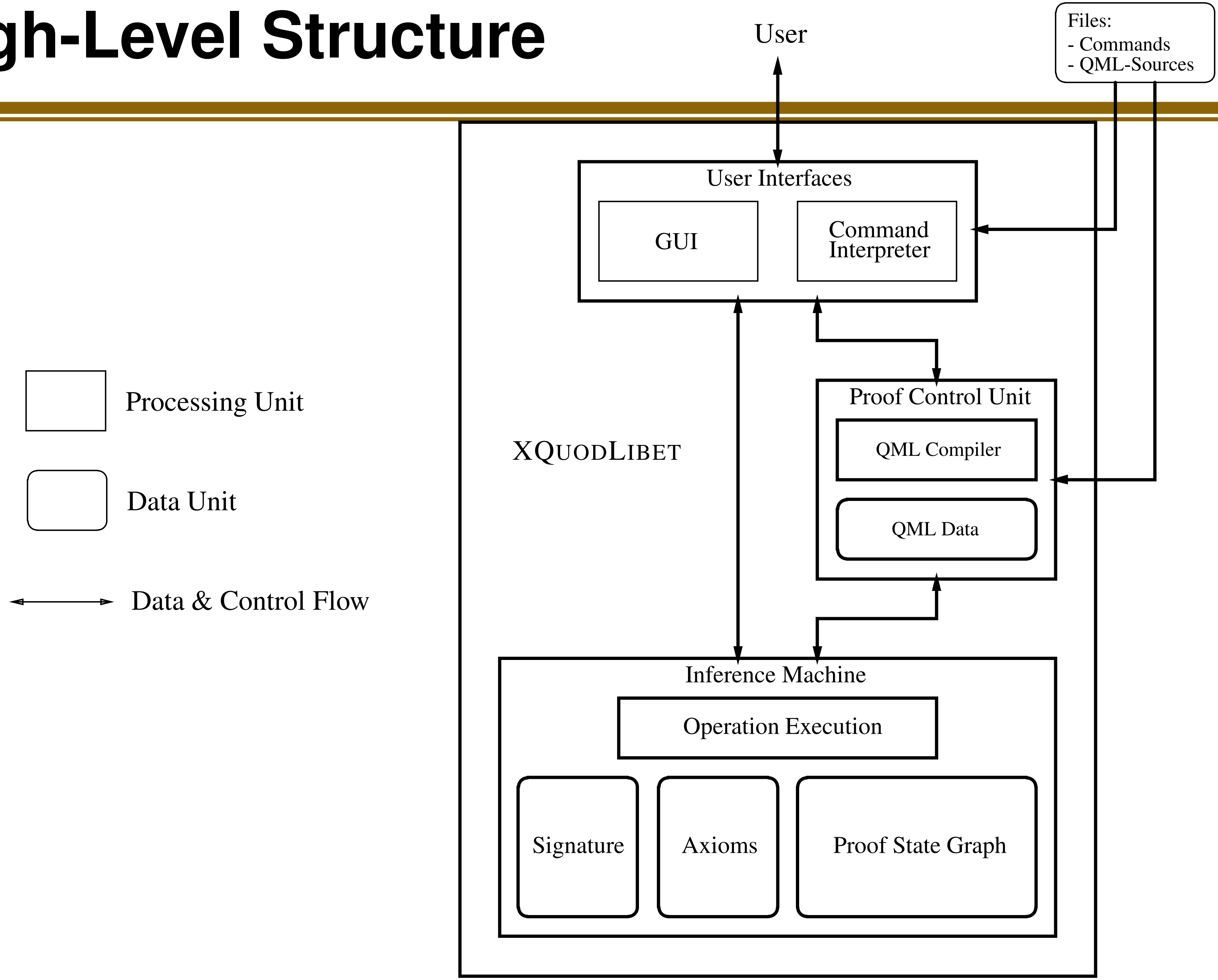
The user can force the system to follow exactly his proof plan by using it as proof checker and then start tactics again.

Why another Inductive Theorem Prover?

- Overcome limitations of
  Explicit Induction Paradigm (EIP)
- Towards a Working Mathematician's style
  (*Descente Infinie*)
- Graceful degradation when automation fails
- Towards an efficient interplay between
  interaction & automation


XQuodLibet
File Windows QML Miscellanea
QuodLibet Log
[y<--s(y),x<--s(x)]
...
minus-ind-lma
QL[31] call simplify-goal minus-ind-lma
Calling simplify-goal for [1^2] in minus-
Applying =-decomp to [1^2] in minus-ind-l
results in no further subgoals
Call of simplify-goal succeeds
The current PS-tree is minus-ind-lma
The current G-node in minus-ind-lma is [1
Runtime: 0.0 sec.
Done
QL[32]
XQL: PS-Tree minus-ind-lma
Window Inference Rules Tactics Commands
Root GNode
Last GNode
Current GNode
Open G
Current Goal Node
G-node [1:2]
{ minus(0,s(y)) < 0,
less(0,s(y)) = true,
s(y) = 0 } ;
w_minus-ind-lma(0,s(y))
Proof State Tree
subst-add
=-decomp
XQL: PS-Tree div-def
Window Inference Rules Tactics Commands
Root GNode
Last GNode
Current GNode
Next Open GNode
Current Goal Node
G-node root "div-def"
{ def div(x,y),
y = 0 } ;
x
++ solved ++
Proof State Tree
lit-add
axm-rewrite
axm-rewrite
def-decomp
lma-subs
def-decomp
ind-subs
lma-subs
lma-subs
XQL: Proof State Trees
PS-Trees Windows View
Proof State Trees
++ less-def-auto
++ minus-def
+ div-def
minus-ind-lma
minus-ind-lma (not solved):
{ minus(x,y) < x,
less(x,y) = true,
y = 0 }
XQL: Defining Rules
Defining Rules
less-1
less-2
less-3
minus-1
minus-2
div-1
div-2
div(x,y) = s(div(minus(x,y),y))
if
y =/= 0,
less(x,y) =/= true,
def less(x,y)
Add Defining Rule
Delete Defining Rule

High-Level Structure



Current Limitations

Accessibility

- $\Longrightarrow$ User-Manual
- $\Longrightarrow$ Agent-based suggestion system (like ΩANTS in ΩMEGA).

Usability

- $\Longrightarrow$ More flexible axiom & lemma activation
- Ca. 5000 lines QML $\Longrightarrow$ New set of tactics, incl. Automatic Generalisation
- $\Longrightarrow$ Reasoning modulo commutativity &c., incl. infix notation
- Only universal variables (parameters)
  $\Longrightarrow$ Existential variables (meta-variables) additionally
- Sequents of first-order literals (clauses)
  $\Longrightarrow$ Sequents of arbitrary higher-order (modal) logic formulas
- Weights provide flexibility for fixed induction order
  $\Longrightarrow$ Also variable induction orderings

Efficiency

- $\Longrightarrow$ Built-in numbers and decision algorithms
- $\Longrightarrow$ $\mathsf{ack}(4, 2) = ?$

Conclusion

- QUODLIBET opens new doors in user-guided automated inductive theorem proving.
- Please come to our system demonstration:

  Computer room next door,
  Saturday 2nd August,
  11:15 h – 11:55 h.