
The QUODLIBET Project

Facts, History, and Future

Claus-Peter Wirth

What It Is

QUODLIBET is an Inductive Theorem Proving (ITP) software system

- designed to support verification and synthesis of non-trivial algorithms

What It Is

QUODLIBET is an Inductive Theorem Proving (ITP) software system

- designed to support verification and synthesis of non-trivial algorithms
- tactic-based

What It Is

QUODLIBET is an Inductive Theorem Proving (ITP) software system

- designed to support verification and synthesis of non-trivial algorithms
- tactic-based
- built-in linear arithmetic (Presburger)

What It Is

QUODLIBET is an Inductive Theorem Proving (ITP) software system

- designed to support verification and synthesis of non-trivial algorithms
- tactic-based
- built-in linear arithmetic (Presburger)
- human-oriented

Why is QUODLIBET Interesting?

Aspects superior to other ITP systems:

- Human-Orientedness
(*Descente Infinie* + Mutual Prompting + GUI)

Why is QUODLIBET Interesting?

Aspects superior to other ITP systems:

- Human-Orientedness
(*Descente Infinie* + Mutual Prompting + GUI)
- Partial and Incremental Definition
(Monotonicity w.r.t. Consistent Extension)

Why is QUODLIBET Interesting?

Aspects superior to other ITP systems:

- Human-Orientedness
(*Descente Infinie* + Mutual Prompting + GUI)
- Partial and Incremental Definition
(Monotonicity w.r.t. Consistent Extension)
- Search Space Organization + Proof Re-Use

Why is QUODLIBET Interesting?

Aspects superior to other ITP systems:

- Human-Orientedness
(*Descente Infinie* + Mutual Prompting + GUI)
- Partial and Incremental Definition
(Monotonicity w.r.t. Consistent Extension)
- Search Space Organization + Proof Re-Use

Competing Systems

(Explicit Induction only, but no *Descente Infinie*):

Why is QUODLIBET Interesting?

Aspects superior to other ITP systems:

- Human-Orientedness
(*Descente Infinie* + Mutual Prompting + GUI)
- Partial and Incremental Definition
(Monotonicity w.r.t. Consistent Extension)
- Search Space Organization + Proof Re-Use

Competing Systems

(Explicit Induction only, but no *Descente Infinie*):

- ACL2 (successor of NQTHM) is so fast that it is suitable for testing in addition to proving. No GUI. No partial specification.

Why is QUODLIBET Interesting?

Aspects superior to other ITP systems:

- Human-Orientedness
(*Descente Infinie* + Mutual Prompting + GUI)
- Partial and Incremental Definition
(Monotonicity w.r.t. Consistent Extension)
- Search Space Organization + Proof Re-Use

Competing Systems

(Explicit Induction only, but no *Descente Infinie*):

- ACL2 (successor of NQTHM) is so fast that it is suitable for testing in addition to proving. No GUI. No partial specification.
- PVS.

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

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]

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

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

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

Contributors

- Initiators

Jürgen Avenhaus Univ. Kaiserslautern

Klaus Madlener Univ. Kaiserslautern

Bernhard Gramlich TU Vienna

Contributors

■ Initiators

Jürgen Avenhaus Univ. Kaiserslautern

Klaus Madlener Univ. Kaiserslautern

Bernhard Gramlich TU Vienna

■ Development Staff

Ulrich Kühler 1990–2000 sd&m AG

Tobias Schmidt-Samoa 1998–2006 ESG GmbH

Claus-Peter Wirth 1990–2008 Saarland Univ.

+ half a dozen of M.Sc. students

Contributors

■ Initiators

Jürgen Avenhaus Univ. Kaiserslautern

Klaus Madlener Univ. Kaiserslautern

Bernhard Gramlich TU Vienna

■ Development Staff

Ulrich Kühler 1990–2000 sd&m AG

Tobias Schmidt-Samoa 1998–2006 ESG GmbH

Claus-Peter Wirth 1990–2008 Saarland Univ.

+ half a dozen of M.Sc. students

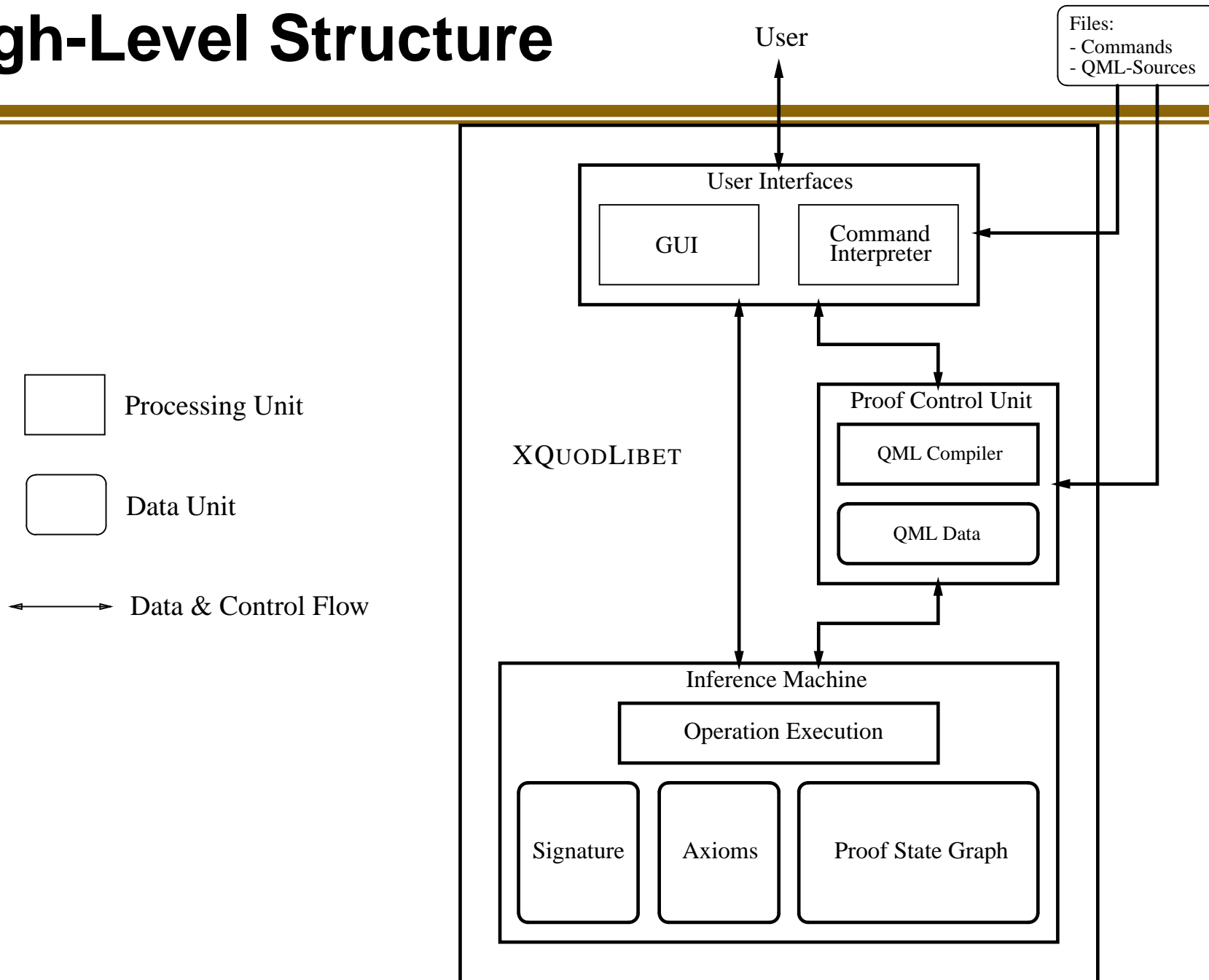
■ User

Bernd Löchner Zühlke Engin. GmbH

History and Future Plans

1990–1995	Requirements Analysis	Kühler, Wirth
1996	Specification	Kühler, Wirth
1996–2008	Overall Design and Coord.	Wirth
1997–2000	Implement. Vers. 1.1	Kühler, M.Sc.s
1998–2000	Termination Comp.	Kühler
1998–2000	Meta-language	Kühler
2003–2004	Implement. Vers. 1.2	Schmidt-Samoa, M.Sc.s
2004–2005	Markings + Presburger	Schmidt-Samoa
2005–2006	Proof Re-Use	Schmidt-Samoa
2005–2006	Implement. Vers. 1.3	Schmidt-Samoa, M.Sc.s
2005–2006	Case Study: LPO	Löchner, Schmidt-Samoa
<hr/>		
2008–2009	Co-Inductive Data Types	Wirth
2009	Implement. Vers. 2.0	Wirth
2009–2010	Rippling, Program Synth.	Wirth, Edinburgh staff

High-Level Structure



Code

Module	Language	Size in KB
{ Inference Machine + Command Interpreter }	COMMON LISP	60
SE/Visibility (MIL graphs)	COMMON LISP	532
QML Compiler	COMMON LISP	713
Heuristics	QUODLIBET ML	1839
GUI	TCL/TK	313

Selected Publications on QUODLIBET

* Claus-Peter Wirth (2008). *Shallow Confluence of Conditional Term Rewriting Systems*. Accepted by J. Symbolic Computation, 39 pp..

Tobias Schmidt-Samoa (2006). *Flexible Heuristic Control for Combining Automation and User-Interaction in Inductive Theorem Proving*. Ph.D. thesis, Univ. Kaiserslautern.

* Tobias Schmidt-Samoa (2006). *Flexible Heuristics for Simplification with Conditional Lemmas by Marking Formulas as Forbidden, Mandatory, Obligatory, and Generous*. J. Applied Non-Classical Logics **16**(1–2), pp. 209–239.

Tobias Schmidt-Samoa (2006). *An Even Closer Integration of Linear Arithmetic into Inductive Theorem Proving*. Electronic Notes in Theoretical Computer Sci. **151**, pp. 3–20, Elsevier.

Serge Autexier, Christoph Benzmüller, Dominik Dietrich, Andreas Meier, Claus-Peter Wirth (2006). *A Generic Modular Data Structure for Proof Attempts Alternating on Ideas and Granularity*. 4th MKM 2005, LNAI 3863, pp. 126–142, Springer.

Claus-Peter Wirth (2005). *History and Future of Implicit and Inductionless Induction: Beware the old jade and the zombie!*. In: Dieter Hutter, Werner Stephan (eds.). *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg Siekmann on the Occasion of His 60th Birthday*, LNAI 2605, pp. 192–203, Springer.

* Claus-Peter Wirth (2004). *Descente Infinie + Deduction*. Logic J. of the IGPL **12**, pp. 1–96, Oxford Univ. Press.

Jürgen Avenhaus, Ulrich Kühler, Tobias Schmidt-Samoa, Claus-Peter Wirth (2003). *How to Prove Inductive Theorems? QUODLIBET!*. 19th CADE 2003, LNAI 2741, pp. 328–333, Springer.

Selected Publications on QUODLIBET (contd.)

- Ulrich Kühler (2000). *A Tactic-Based Inductive Theorem Prover for Data Types with Partial Operations*. Ph.D. thesis, Infix, Akademische Verlagsgesellschaft Aka GmbH, Sankt Augustin, Berlin.
- Claus-Peter Wirth (2000). *Full First-Order Sequent and Tableau Calculi With Preservation of Solutions and the Liberalized δ -Rule but Without Skolemization*. In: Ricardo Caferra, Gernot Salzer (eds.). *Automated Deduction in Classical and Non-Classical Logics*, LNAI 1761, pp. 283–298, Springer.
- Claus-Peter Wirth (1999). *Full First-Order Free Variable Sequents and Tableaus in Implicit Induction*. 8th TABLEAUX 1999, LNAI 1617, pp. 293–307, Springer.
- Ulrich Kühler, Claus-Peter Wirth (1997). *Conditional Equational Specifications of Data Types with Partial Operations for Inductive Theorem Proving*. 8th RTA 1997, LNCS 1232, pp. 38–52, Springer.
- Claus-Peter Wirth (1997). *Positive/Negative-Conditional Equations: A Constructor-Based Framework for Specification and Inductive Theorem Proving*. Ph.D. thesis, 256 pp., Verlag Dr. Kovač, Hamburg.
- Claus-Peter Wirth, Bernhard Gramlich (1994). *On Notions of Inductive Validity for First-Order Equational Clauses*. 12th CADE 1994, LNAI 814, pp. 162–176, Springer.
- * Claus-Peter Wirth, Bernhard Gramlich (1994). *A Constructor-Based Approach for Positive/Negative-Conditional Equational Specifications*. *J. Symbolic Computation* **17**, pp. 51–90, Academic Press (Elsevier).