System Description

Analytica 2

joint work with

Edmund Clarke, Joël Ouaknine, Klaus Sutner

Michael Kohlhase
School of Engineering & Science
International University Bremen, Germany

http://www.cs.cmu.edu/~kohlhase

 ©: Michael Kohlhase

ANALYTICA (recap)

- ANALYTICA is a theorem proving system for 19^{th} century mathematics
  - summation formulae (Ramanujan's notebooks)
  - calculus (e.g. Bernstein approximation theorem)
- written on top of the MATHEMATICA computer algebra system
  - developed in the early 1990's by Xudong Zhao and Edmund Clarke
  - has since been dormant.
- under the hood:
  - sequent tableau calculus (Proof transcripts as side-effects)
  - strong simplification (MATHEMATICA simplifier + domain-specific rules)
  - algebraic simplification during unification (completeness?)
- Summary: brilliant tour de force (what can we learn from this?)

©: Michael Kohlhase

Analytica and Mathematica (recap)

- Market leading Computer Algebra System (CAS) (proprietary: Wolfram Inc.)
  - Kernel: Symbolic math routines + Programming Language
  - Notebook Front-End: Formula editor + folding + interaction
  - External Systems Interface: JLink runs Java, exports CAS functions
  - Native Xml interface: interprets Xml trees as symbolic expressions
- Mathematica programming language: Higher-Order Conditional Rewriting

```
Imply[seq[h_, or[c1___, and[a_, b__], c2___]]] :=
            (print["and split"];
             SequentialTry[ seq[h, or[c1, a, c2]],
                            seq[and[h, simple[a]],
                            or[c1, and[b], c2]]]);
```

  - like logic programming in style
  - sequence variables: __ (non-empty) and ___ (possibly empty)
- Analytica was originally developed for Mathematica 1.2

©: Michael Kohlhase

Porting ANALYTICA to MATHEMATICA 5

- transform the ~ 50 source files into two notebooks (prover and knowledge)
  - document the prover code in special SOURCE STYLE (literate programming)
  - generate program + MATHEMATICA help files from that
- Example: Weak simplification

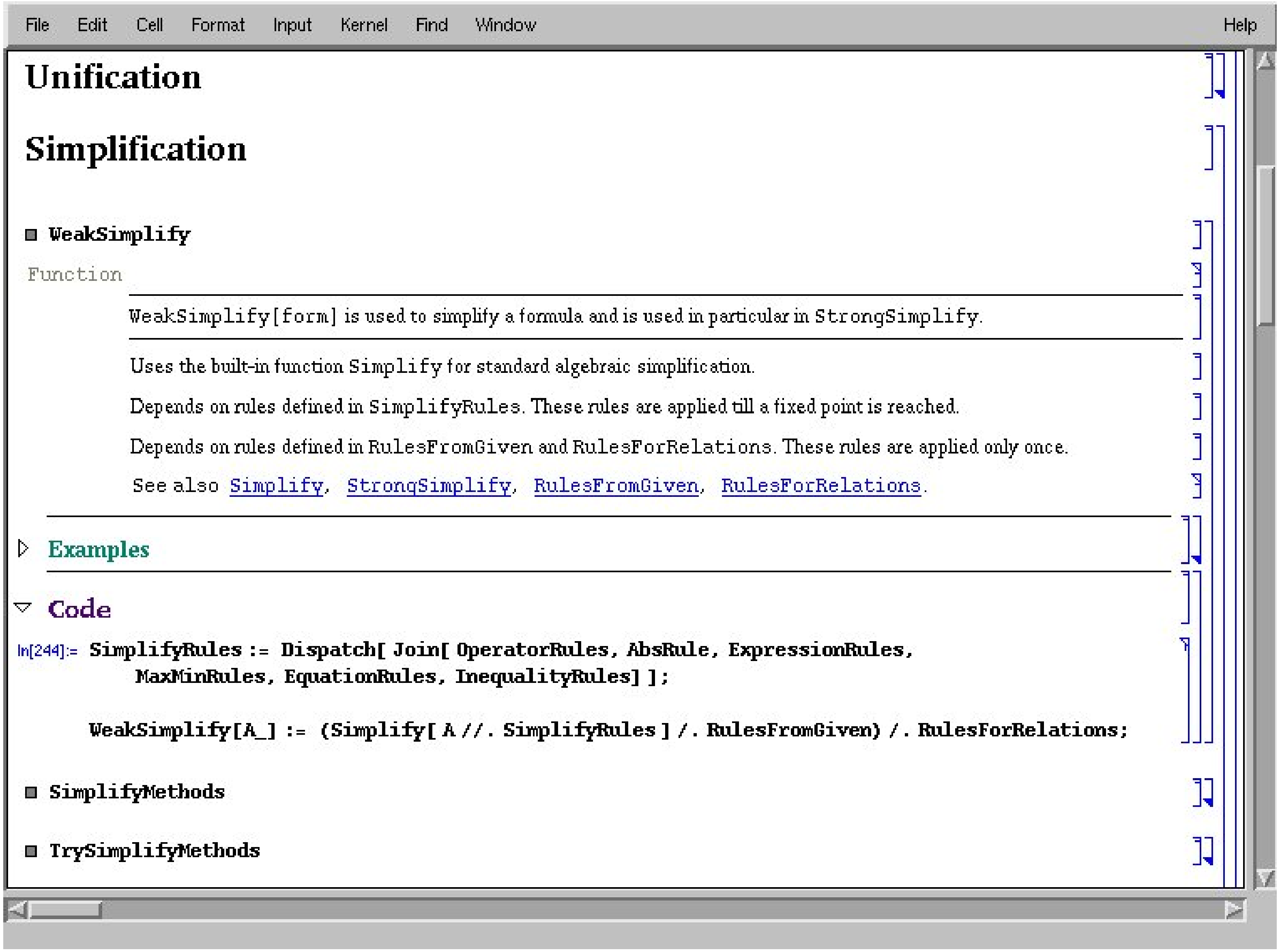
```
SimplifyRules := Join[OperatorRules, AbsRule, ExpressionRules,
                      MaxMinRules, EquationRules, InequalityRules];

(* WeakSimplify is used to simplify sub-formula and
                is a tactic in StrongSimplify. *)

WeakSimplify[f_] :=
        Simplify[f //. SimplifyRules] /.
        RulesFromGiven /. RulesForRelations;
```


©: Michael Kohlhase

Weak Simplification in the Notebook



File Edit Cell Format Input Kernel Find Window Help

Unification

Simplification

WeakSimplify

Function

WeakSimplify[form] is used to simplify a formula and is used in particular in StrongSimplify.

Uses the built-in function Simplify for standard algebraic simplification.

Depends on rules defined in SimplifyRules. These rules are applied till a fixed point is reached.

Depends on rules defined in RulesFromGiven and RulesForRelations. These rules are applied only once.

See also Simplify, StrongSimplify, RulesFromGiven, RulesForRelations.

Examples

Code

```
In[244]:= SimplifyRules := Dispatch[ Join[ OperatorRules, AbsRule, ExpressionRules,
              MaxMinRules, EquationRules, InequalityRules] ];

          WeakSimplify[A_] := (Simplify[ A //. SimplifyRules ] /. RulesFromGiven) /. RulesForRelations;
```

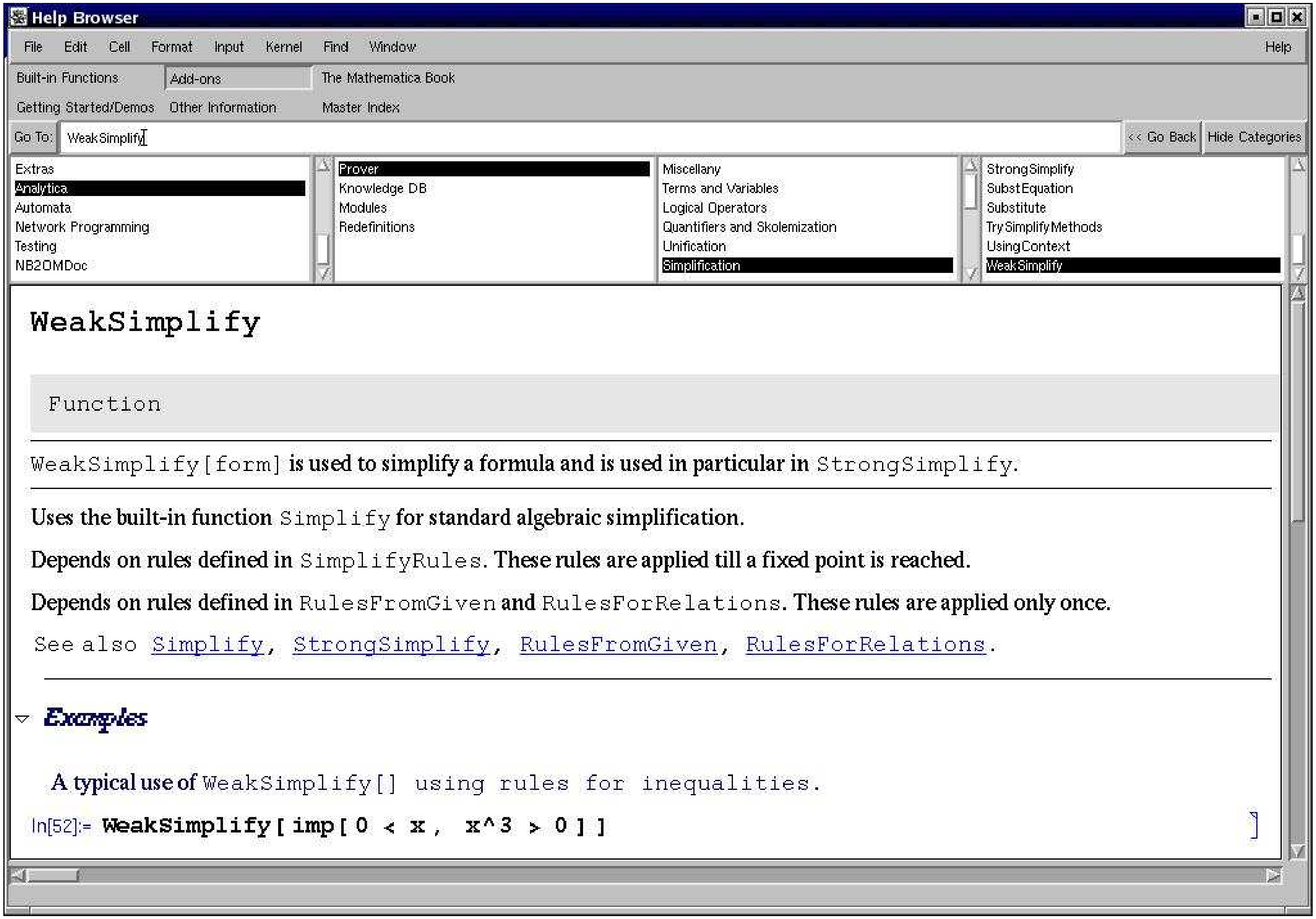
SimplifyMethods

TrySimplifyMethods

©: Michael Kohlhase

IU^{B})

The corresponding Entry in the Help browser



©: Michael Kohlhase

Separating Mathematical Knowledge from Code

- Not all ANALYTICA code is computational

```
(* Rules for simplifying expressions involving abs. *)
Abs[a_ b_] := Abs[a] Abs[b];
Abs[a_^n_] := Abs[a]^n;

(*  Local rule  used in simplification. *)
AbsRule = {Abs[a_] :> If[TrueQ[WeakSimplify[a >= 0]], a,
                 If[TrueQ[WeakSimplify[a <= 0]], -a,
                         Abs[Factor1[a]]]]};
```

- mathematical equivalent

#	formalization	the absolute value function ...
1	$\forall a, b. \lvert a \cdot b \rvert = \lvert a \rvert \cdot \lvert b \rvert$	commutes with multiplication
2	$\forall a, n. \lvert a^n \rvert = \lvert a \rvert^n$	commutes with exponentiation
3	$\forall a. a \geq 0 \Rightarrow \lvert a \rvert = a$	is the identity on $\mathbb{R}^+$
4	$\forall a. a \leq 0 \Rightarrow \lvert a \rvert = -a$	is the negative identity on $\mathbb{R}^-$

©: Michael Kohlhase

Separating Mathematical Knowledge from Code (Howto)

- ANALYTICA knowledge: symbol declarations + rewriting rules
  - used directly by the MATHEMATICA simplifier (first block)
  - used in Weak simplification procedure discussed above (second block)
- The soundness of ANALYTICA depends on hundreds of such rules
- Problem: how to make this explicit and manage it.
- Idea: use a mathematical knowledge representation system
- OMDOC: Open Mathematical Documents ([Kohlhase '99-'03])
  - convert using the `nb2omdoc` utility ([Sutner '03], using MATHEMATICA's native XML integration)
  - document as mathematical theorems in vernacular and logic
  - split into relevant mathematical theories
  - store in mathematical knowledge base system MBASE.

OMDoc in a Nutshell (three levels of modeling)

Formula level: OpenMath/C-MathML

- Objects as logical formulae
- semantics by ref. to theory level

```
<OMA>
 <OMS cd="arith1" name="plus"/>
 <OMS cd="nat" name="zero"/>
 <OMV name="N"/>
</OMA>
```

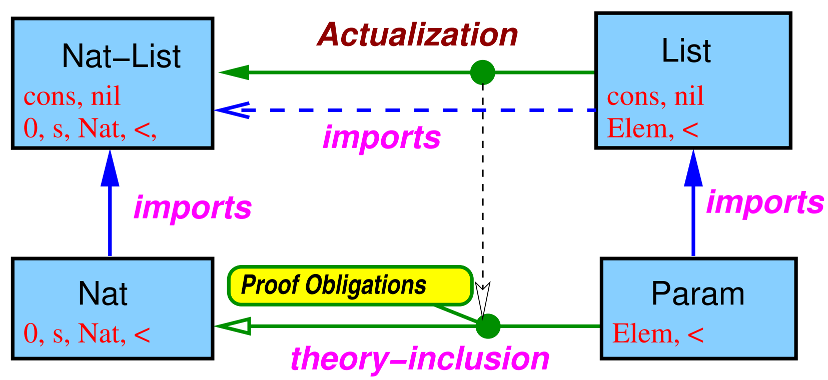
Statement level:

- Definition, Theorem, Proof, Example
- semantics explicit forms and refs.

```
<defn for="plus" type="rec">
 <CMP>rec. eq. for plus</CMP>
 <FMP>X+0 = 0</FMP>
 <FMP>X+s(Y) = s(X+Y)</FMP>
</defn>
```

Theory level: Development Graph

- inheritance via symbol-mapping
- theory-inclusion by proof-obligations
- local (one-step) vs. global links



©: Michael Kohlhase

Problems with the translation

- Idiosyncrasies of the representation ($\leadsto$ massage away!)
- Sequence Variables, polyadic functions (what is the logic? [Kutsia '02])

```
Continuous[f_[a__], x_, x0_] :=
    Apply[ and, Map[ Function[z, Continuous[z, x, x0]], List[a]]] /; ContFunction[f];
```

- Idea: treat sequence variables as arbitrary variables and represent this as

$$\forall f, a, x, x_0.\mathbb{C}(f(a), x, x_0) \Leftrightarrow \mathbb{C}^0(f) \wedge apply(\wedge, map(\lambda z \mathbb{C}(z, x, x_0))), list(a)$$

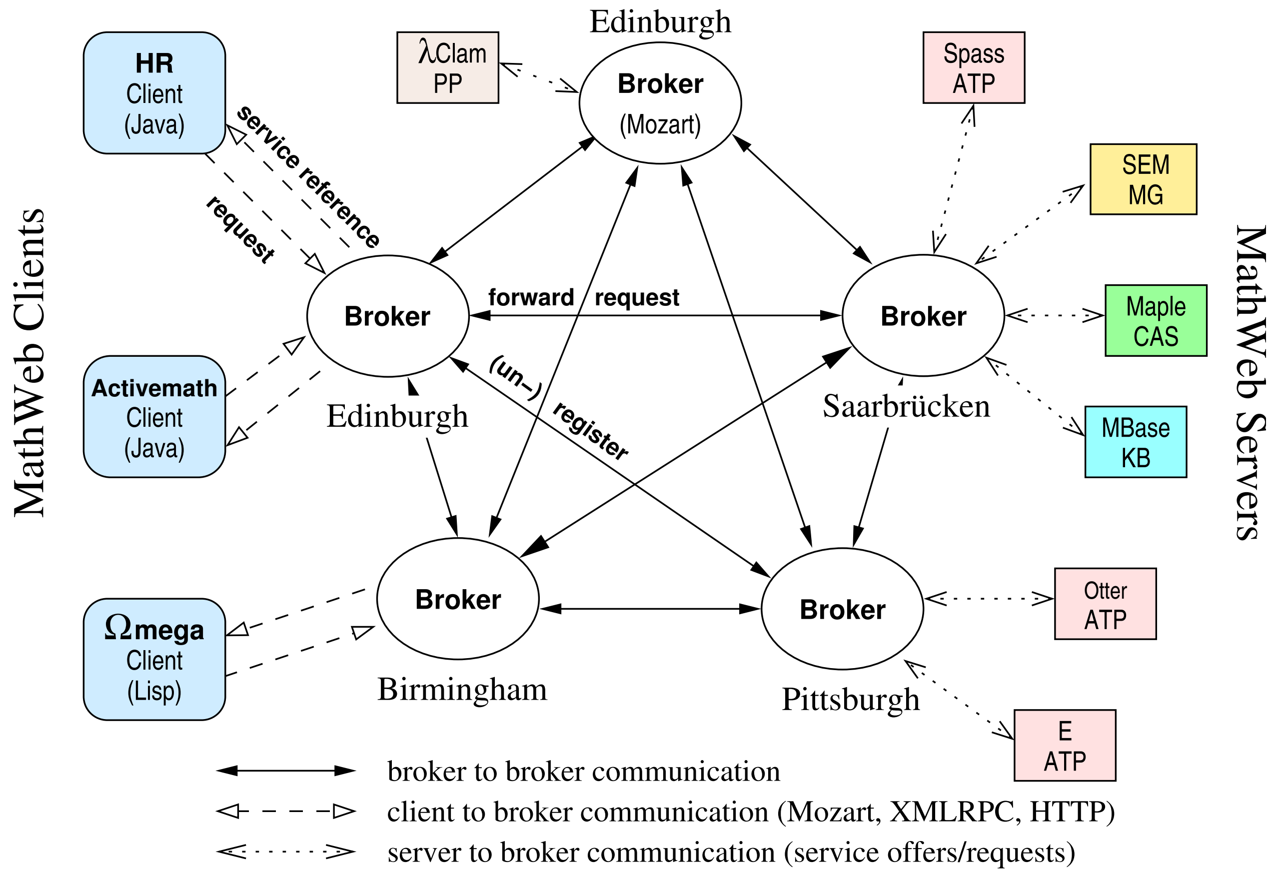
- Problem: How to communicate with other theorem provers?
- Idea: use higher-order logic with Currying (e.g. for TPS)

  $\forall F, A, x, x_0.\mathbb{C}((F A), x, x_0) \Leftrightarrow \mathbb{C}(F, x, x_0) \wedge \mathbb{C}(A, x, x_0)$ and
  $\forall F, x, x_0.\mathbb{C}(F, x, x_0) \Leftarrow \mathbb{C}^0(F)$.
- Example: prove $\mathbb{C}(x^2 + x, x, 1)$ (trace ANALYTICA computation)

  $\mathbb{C}^0(+) \leadsto \mathbb{C}(+, x, 1)$ and $\mathbb{C}(x, x, x_0) \leadsto \mathbb{C}(x^2, x, x_0)$,
  thus $\mathbb{C}(+x^2, x, x_0)$, and finally $\mathbb{C}(x^2 + x, x, 1)$

©: Michael Kohlhase
IU^B)

A MathWeb Interface for Analytica



- equip Analytica with an Xml-Rpc interface via JLink
- use native OMDoc transformation for communication with other services

©: Michael Kohlhase

IU^B)

Communicating with MBase via Xml-Rpc

```
<methodCall><methodName>Broker.getService</methodName>
 <params><param><value><string>MBase</string></value></param></params>
</methodCall>

<methodResponse><methodBody>http://mbase.mathweb.org:12345</methodBody></methodResponse>

<methodCall><methodName>provide.theory.trans</methodName>
 <params><param><struct>
  <member><name>1</name><value><string>contfunc</string></value></member>
   <member><name>format</name><value><string>OMDoc</string></value></member>
  </struct></param></params>
</methodCall>

<methodResponse><methodBody>
  <theory id="contfunc">
    <imports id="i0" from="integer"/>
    <symbol id="continuous">
      <CMP>  C(f(a), x, x_0) is True when f is continuous at point x_0.</CMP>
    </symbol>
    <assertion id="CNTF1"><CMP>The identity function is continuous.</CMP></assertion>
    <code id="c2" pto="Mathematica" pto-version="4.2" for="CNTF1">
      <data><![CDATA[EVALUATE[Continuous[x_, x_, z_] := True; ]]]></data>
    </code>
    ...
</methodBody></methodResponse>
```

 ©: Michael Kohlhase IU^B

Implementing a MathWeb interface $\rightsquigarrow$ JLink

- Implementing a MathWeb client (accessing MBase from Analytica)
  - call standard Java Xml-Rpc package via JLink (apache.org)
  - import result into Mathematica as symbolic Xml (CAS terms for Xml tree)
  - manipulate with CAS functions (extract Mathematicacode, evaluate)
  - implementation in one afternoon (if you have the theories)
- Implementing an Analytica server (work in progress)
  - run standard Xml-Rpc package in Java process
  - upon request, start Mathematica kernel via JLink (niced, timeout,...)
  - provide request to Analytica as symbolic Xml (need OMDoc parser [Sutner '02])
  - load necessary background theories from MBase
  - return OMDoc/Xml proof to Java, which answers Xml-Rpc request

©: Michael Kohlhase

Conclusions and Further Work

- ANALYTICA 2: still a theorem prover for 19^{th} mathematics
  - implemented in MATHEMATICA like THEOREMA
  - ultra-tight interaction of CAS and Ded (correctness?)
  - new technologies under the hood (Notebooks and XML)
- Status: Ongoing experiment to resurrect (and understand) ANALYTICA
  - unusual software platform, symbiosis with CAS, symbolic periphery
- Future: start extending the system
  - Proof objects, (extract them; flag oracles, theory dependencies)
  - more Math(ematica) (use the extra power in M5: e.g. Gosper's alg.)
  - MATHWEB service (have client (MBASE))
  - ADV: looking for Ph.D students for this (CMU or IUB)

©: Michael Kohlhase