Base Technologies for Tutoring
Experiences from experiments in the ISAC-project

Walther Neuper
Institute for Softwaretechnology
Graz University of Technology

Time Conference 2010
Jul.8, Malaga
Outline

1. Remarks on tutoring software
   Status quo in the design
   Requirements on the design

2. Base technologies for tutoring
   Computer theorem proving
   Single-stepping program interpretation
   Human readable math knowledge

3. Summary and Conclusion
1 Remarks on tutoring software
   Status quo in the design
   Requirements on the design

2 Base technologies for tutoring
   Computer theorem proving
   Single-stepping program interpretation
   Human readable math knowledge

3 Summary and Conclusion
Status quo in tutoring software

Software for tutoring is concerned with *individuals* . . .
- . . . *individual learners*
  - on different levels
  - with different pace in learning . . .
- . . . *individual teachers*
  - with different teaching styles
  - emphasizing specific examples . . .
- . . . *individual programmers* (frequently teachers)
  - creating an abundant variety of software
  - lack support of general software services

What are the *general requirements* for tutoring ?
What are the *basic technologies* to meet the requirements ?
Status quo in tutoring software

Software for tutoring is concerned with *individuals* . . .

- *individual learners*
  - on different levels
  - with different pace in learning . . .

- *individual teachers*
  - with different teaching styles
  - emphasizing specific examples . . .

- *individual programmers* (frequently teachers)
  - creating an abundant variety of software
  - lack support of general software services

What are the **general requirements** for tutoring?
What are the **basic technologies** to meet the requirements?
Status quo in tutoring software

Software for tutoring is concerned with individuals ...  

- ... *individual learners*  
  - on different levels  
  - with different pace in learning ...  

- ... *individual teachers*  
  - with different teaching styles  
  - emphasizing specific examples ...  

- ... *individual programmers* (frequently teachers)  
  - creating an abundant variety of software  
  - lack support of general software services

What are the *general requirements* for tutoring?  
What are the *basic technologies* to meet the requirements?


1. Remarks on tutoring software
   Status quo in the design
   Requirements on the design

2. Base technologies for tutoring
   Computer theorem proving
   Single-stepping program interpretation
   Human readable math knowledge

3. Summary and Conclusion
Requirements for tutoring

The general requirements are basically:

1. **Check user input** as generous as possible . . .

2. **Guide the user** step by step towards a solution . . .

3. **Explain steps** on request by the user . . .

. . . during the stepwise *construction* of the solution of some problem in applying mathematics (incl. geometry).
Requirements for tutoring

The general requirements are basically:

1. **Check user input** as generous as possible . . .

2. **Guide the user** step by step towards a solution . . .

3. **Explain steps** on request by the user . . .

. . . during the stepwise *construction* of the solution of some problem in applying mathematics (incl. geometry).
Requirements for tutoring

The general requirements are basically:

1. **Check user input** as generous as possible . . .
2. **Guide the user** step by step towards a solution . . .
3. **Explain steps** on request by the user . . .

. . . during the stepwise *construction* of the solution of some problem in applying mathematics (incl. geometry).
Requirements for tutoring

The general requirements are basically:

1. **Check user input** as generous as possible . . .

2. **Guide the user** step by step towards a solution . . .

3. **Explain steps** on request by the user . . .

...during the stepwise *construction* of the solution of some problem in applying mathematics (incl. geometry).
Requirements for tutoring

The general requirements are basically:

1. **Check user input** as generous as possible . . .
2. **Guide the user** step by step towards a solution . . .
3. **Explain steps** on request by the user . . .

. . . during the stepwise *construction* of the solution of some problem in applying mathematics (incl. geometry).
Remarks on tutoring software
Status quo in the design
Requirements on the design

Base technologies for tutoring
Computer theorem proving
Single-stepping program interpretation
Human readable math knowledge

Summary and Conclusion
Requirements and Technologies

The requirements can be met by the technologies

1. **check user input** as generous and liberal as possible
   - by Computer Theorem Proving (CTP), i.e.
   - Automated Proving (ATP): simplifiers, SAT, SMT etc
   - Interactive Proving (ITP): Coq and Isabelle in EU

2. **guide the user** step by step towards a solution

3. **explain steps** on request by the user
Requirements and Technologies

The requirements can be met by the technologies

1. check user input as generous and liberal as possible
   - by Computer Theorem Proving (CTP), i.e.
   - Automated Proving (ATP): simplifiers, SAT, SMT etc
   - Interactive Proving (ITP): Coq and Isabelle in EU

2. guide the user step by step towards a solution

3. explain steps on request by the user
Requirements and Technologies

The requirements can be met by the **technologies**

1. **check user input** as generous and liberal as possible
   - by **Computer Theorem Proving (CTP)**, i.e.
   - **Automated Proving (ATP)**: simplifiers, SAT, SMT etc
   - **Interactive Proving (ITP)**: Coq and Isabelle in EU

2. **guide the user** step by step towards a solution

3. **explain steps** on request by the user
Requirements and Technologies

The requirements can be met by the *technologies*

1. **check user input** as generous and liberal as possible
   - by *Computer Theorem Proving (CTP)*, i.e.
   - *Automated Proving (ATP)*: simplifiers, SAT, SMT etc
   - *Interactive Proving (ITP)*: Coq and Isabelle in EU

2. **guide the user** step by step towards a solution
   - 
   - 

3. **explain steps** on request by the user
   - 
   -
Demo Isabelle from wikipedia


1 Remarks on tutoring software
   Status quo in the design
   Requirements on the design

2 Base technologies for tutoring
   Computer theorem proving
   Single-stepping program interpretation
   Human readable math knowledge

3 Summary and Conclusion
Requirements and Technologies

The requirements can be met by the technologies

1. check user input as generous and liberal as possible
   - by Computer Theorem Proving (CTP), i.e
   - Automated Proving (ATP): simplifiers, SAT, SMT etc
   - Interactive Proving (ITP): Coq and Isabelle in EU

2. guide the user step by step towards a solution
   - by interpretation in debug-mode with breakpoints

3. explain steps on request by the user
Requirements and Technologies

The requirements can be met by the **technologies**

1. **check user input** as generous and liberal as possible
   - by Computer Theorem Proving (CTP), i.e.
   - Automated Proving (ATP): simplifiers, SAT, SMT etc
   - Interactive Proving (ITP): Coq and Isabelle in EU

2. **guide the user** step by step towards a solution
   - by **interpretation in debug-mode** with breakpoints

3. **explain steps** on request by the user
Interpretation in debug-mode

Worksheet

Dialog module

Tutor

Isabelle/Isar

Logical operating system (contexts etc)

Interpreter (P.L)

Specification

In: function q,
   Length L
pre: q is integrable
   ∆ L > 0
out: function y(x)
Post: y(0) = 0
   ∆ y'(0) = 0
   ∆ V(0) = q.L
   ∆ M_b(L) = 0

Program

Script B (q, L, v, Cs) =
   LET
      funs = Subproblem
      (thy, pbl, met) q, L, v
      equus = Subproblem ...
sols = Subproblem ...
   B = Take (LAST funs)
   B = (Substitute sols)@
      (Rewrite_Set poly)) B
   IN B

Output

Authoring
Example: start tutoring

Tutoring

technology for tutoring
walther neuper

status quo
requirements

technologies
ctp
debug mode
transparent

summary and conclusion

worksheet
dialog module

isabelle/isar
logical operating system
(contexts etc)

interpreter (P.L)

specification

program

output

Tutoring

worksheet

dialog module

isabelle/isar

logical operating system

(contexts etc)

interpreter (P.L)

specification

program

output

In: function q, Length L
pre: q is_integrable
\[ \Delta L > 0 \]
out: function y(x)
Post: y(0)=0 \[ \Delta y'(0)=0 \]
\[ \Delta V(0)=q.L \]
\[ \Delta M_b(L)=0 \]

Script B (q, L, v, Cs) =
LET
  funs = Subproblem
  equs = Subproblem ...
  sols = Subproblem ...
  B = Take (LAST funs)
B = ((Substitute sols)@ (Rewrite_Set poly)) B
IN B

In: function q, Length L
pre: q is_integrable
\[ \Delta L > 0 \]
out: function y(x)
Post: y(0)=0 \[ \Delta y'(0)=0 \]
\[ \Delta V(0)=q.L \]
\[ \Delta M_b(L)=0 \]
Example: Tutoring start

Tutoring 

Problem (B, bendl) 

worksheet
Problem (B, bendl)

dialog module

Tutor

Isabelle/Isar
logical operating system
(contexts etc)

Interpreter (P.L)

specification

In: function q,
Length L
pre: q is_integrable
\( \Delta L > 0 \)
out: function y(x)
Post: \( y(0)=0 \), \( \Delta y'(0)=0 \)
\( \Delta V(0)=q.L \)
\( \Delta M_b(L)=0 \)

program

Script B (q, L, v, Cs) = 
LET
funs = Subproblem
(thy, pbl, met) q, L, v
equs = Subproblem ...
sols = Subproblem ...
B = Take (LAST funs)
B = (Substitute sols)@
(Rewrite_Set poly)) B
IN B

output
Example: Tutoring step 1

The diagram illustrates the flow of a tutoring system using Isabelle/Isar as the logical operating system. The tutoring process involves the following components:

- **Worksheet**: Contains the problem statement and initial conditions.
- **Dialog Module**: Facilitates the interaction between the tutor and learner.
- **Isabelle/Isar**: The logical operating system with context handling and specification and program generation.
- **Interpreter (P.L)**: Processes the generated program output.

The tutor provides a logical specification of the problem, which is then interpreted and executed. The tutor module receives feedback based on the learner's interactions and progress.

### Logical Specification

```
IN:  function q, Length L
pre: q is_integrable  Δ L > 0
out: function y(x)
Post: y(0)=0 Δ y'(0)=0
      Δ V(0)=q.L  Δ M_b(L)=0

Script B (q, L, v, Cs) =
  LET
  funs = Subproblem (thy, pbl, met) q, L, v 
equs = Subproblem ...
sols = Subproblem ...
B = Take (LAST funs)
B = ((Substitute sols)@ (Rewrite_Set poly)) B
IN B
```
Example: Tutoring step 2

Tutoring

Problem (B, bendl)
Problem (B, load2bl)
Q(x) = c·q·x, M(x) = ...
Problem (B, sidecds)
L·q = x, 0 = c_2+L·c...

worksheet

In: function q,
Length L
pre: q is_integrable
\Delta L > 0
out: function y(x)
Post: y(0)=0 \Delta y'(0)=0
\Delta V(0)=q·L
\Delta M_b(L)=0

Isabelle/Isar
logical operating system
(contexts etc)

specification

In: function q, Length L
pre: q is_integrable
\Delta L > 0
out: function y(x)
Post: y(0)=0 \Delta y'(0)=0
\Delta V(0)=q·L
\Delta M_b(L)=0

program

Script B (q, L, v, Cs) =
LET
funs = Subproblem
(thy, pbl, met) q, L, v
equs = Subproblem ...
sols = Subproblem ...
B = Take (LAST funs)
B = ((Substitute sols)@ (Rewrite_Set poly)) B
IN B

output

Tutoring
Example: Tutoring step 3

Tutoring

worksheet

Problem (B, bendl)
Problem (B, load2bl)
Q(x) = c·q·x, M(x) = ...
Problem (B, sidecads)
L·q = x, 0 = c_2+L·c...
solveSys [0=c_3, ...
c = q·L, c_2 = -L^2·q/2.

dialog module

Isabelle/Isar
logical operating system
(contexts etc)

specification

In: function q,
Length L

pre: q is integrable
\[ \Delta L > 0 \]

out: function y(x)

Post: y(0)=0 \[ \Delta y'(0)=0 \]
\[ \Delta V(0)=q·L \]
\[ \Delta M_b(L)=0 \]

program

Script B (q, L, v, Cs) = LET
funs = Subproblem
(thy, pbl, met) q, L, v
equ = Subproblem ...
sols = Subproblem ...
B = Take (LAST funs)
B = ((Substitute sols)@(Rewrite_Set poly)) B
IN B

output

Tutoring

Example: Tutoring step 3

Tutoring

worksheet

Problem (B, bendl)
Problem (B, load2bl)
Q(x) = c·q·x, M(x) = ...
Problem (B, sidecads)
L·q = x, 0 = c_2+L·c...
solveSys [0=c_3, ...
c = q·L, c_2 = -L^2·q/2.

dialog module

Isabelle/Isar
logical operating system
(contexts etc)

specification

In: function q,
Length L

pre: q is integrable
\[ \Delta L > 0 \]

out: function y(x)

Post: y(0)=0 \[ \Delta y'(0)=0 \]
\[ \Delta V(0)=q·L \]
\[ \Delta M_b(L)=0 \]

program

Script B (q, L, v, Cs) = LET
funs = Subproblem
(thy, pbl, met) q, L, v
equ = Subproblem ...
sols = Subproblem ...
B = Take (LAST funs)
B = ((Substitute sols)@(Rewrite_Set poly)) B
IN B

output

Tutoring
**Example: Tutoring step 4**

Tutoring

### worksheet
- Problem (B, bendl)
- Problem (B, load2bl)
- Problem (B, sidecds)

```
L.q = x, 0 = c_2+L.c...
solveSys [0=c_3, ...
c = q.L, c_2 = -L^2.q/2.
y(x) = c_4+c_3.x-1/EI...
```

### dialog module

### specification
- In: function q, Length L
- pre: q is integrable
- Post: y(0)=0 \( \Delta y'(0)=0 \)
- Out: function y(x)

```
\Delta L > 0
\Delta V(0) = q.L
\Delta M_b(L) = 0
```

### program
- Script B (q, L, v, Cs) =
  - LET
    - funs = Subproblem
      - (thy, pbl, met) q, L, v
    - equus = Subproblem ...
    - sols = Subproblem ...
  - B = Take (LAST funs)
  - B = ((Substitute sols)@ (Rewrite_Set poly)) B

### output

**Isabelle/Isar**
- logical operating system
  - (contexts etc)

**Interpreter (P.L)**
Example: Tutoring step 5

Tutoring

worksheet
Problem (B, bendl)
Problem (B, load2bl)
Q(x) = c·q·x, M(x) = ...
Problem (B, sidecds)
L·q = x, 0 = c_2 + L·c...
solveSys [0 = c_3, ...
c = q·L, c_2 = -L^2·q/2.
y(x) = c_4 + c_3·x - 1/EI...
y(x) = 0 + 0·x - 1/EI...

dialog module

Isabelle/Isar
logical operating system
(contexts etc)

specification
In: function q,
Length L
pre: q is_integrable
\Delta L > 0
out: function y(x)
Post: y(0) = 0 \Delta y'(0) = 0
\Delta V(0) = q·L
\Delta M_b(L) = 0

program
Script B (q, L, v, Cs) =
LET
funs = Subproblem
(thy, pbl, met) q, L, v
equs = Subproblem ...
sols = Subproblem ...
B = Take (LAST funs)
B = ((Substitute sols)@ (Rewrite_Set poly)) B
IN B

output

Tutoring
Example: Tutoring finished

**Worksheet**

Problem (B, bendl)
Problem (B, load2bl)
Q(x) = c-q.x, M(x) = ...
Problem (B, sideccls)
L.q = x, 0 = c_2+L.c...
solveSys [0=c_3, ...
c = q.L, c_2 = -L^2.q/2.
y(x) = c_4+c_3.x-1/El...
y(x) = 0 + 0.x - 1/El ...
y(x) = (q.L^2)/(4.El) . x^2
- (q.L)/(6.El) . x^3
+ q /(24.El) . x^4

**Dialog Module**

**Isabelle/Isar**

Logical operating system
(contexts etc)

**Interpreter (P.L)**

**Specification**

In: function q,
Length L
pre: q is integrable
Δ L > 0
out: function y(x)
Post: y(0)=0 Δ y'(0)=0
Δ V(0)=q.L
Δ M_b(L)=0

**Program**

Script B (q, L, v, Cs) =
LET
funs = Subproblem
(thy, pbl, met) q, L, v
equs = Subproblem ...
sols = Subproblem ...
B = Take (LAST funs)
B = ((Substitute sols)@
(Rewrite_Set poly)) B
IN B

**Output**

Tutoring
Same problem: just get result

Isabelle/Isar
logical operating system
(contexts etc)

**specification**
In: function q, Length L
pre: q is_integrable
Δ L > 0
out: function y(x)
Post: y(0)=0 Δ y'(0)=0
Δ V(0)=q.L
Δ M_b(L)=0

**program**
Script B (q, L, v, Cs) =
LET
funs = Subproblem
thy, pbl, met) q, L, v
equs = Subproblem …
sols = Subproblem …
B = Take (LAST funs)
B = ((Substitute sols)@
(Rewrite_Set poly)) B
IN B

**output**
y(x) = (q.L^2)/(4.EI) . x^2
- (q.L)/(6.EI) . x^3
+ q /(24.EI) . x^4
Requirements and Technologies

The requirements can be met by the technologies

1. check user input as generous and liberal as possible
   - by Computer Theorem Proving (CTP), i.e.
   - Automated Proving (ATP): simplifiers, SAT, SMT etc
   - Interactive Proving (ITP): Coq and Isabelle in EU

2. guide the user step by step towards a solution
   - by interpretation in debug-mode with breakpoints
   - ? resume after liberal user input at breakpoint (ATP !)

3. explain steps on request by the user
   -
Requirements and Technologies

The requirements can be met by the technologies

1. **check user input** as generous and liberal as possible
   - by Computer Theorem Proving (CTP), i.e.
   - Automated Proving (ATP): simplifiers, SAT, SMT etc
   - Interactive Proving (ITP): Coq and Isabelle in EU

2. **guide the user** step by step towards a solution
   - by *interpretation in debug-mode* with breakpoints
   - ? **resume** after liberal user input at breakpoint (ATP !)

3. **explain steps** on request by the user
   -
Dialog mechanically generated by CTP-technology

A calculation proceeded to a certain step (no. 3.):

1. $\frac{d}{dx}(x^2 + \sin(3 \cdot x^4))$

2. $2 \cdot x^{2-1} + \frac{d}{dx} \sin(3 \cdot x^4)$

3. $2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4)$

How can we guide the student to the next formula (no. 4.)

...  

4. $2 \cdot x + \cos(3 \cdot x^4) \cdot \frac{d}{dx} (3 \cdot x^4)$

... or some algebraically equivalent formula?
Dialog mechanically generated by CTP-technology

A calculation proceeded to a certain step (no. 3.):

1. \( \frac{d}{dx}(x^2 + \sin(3 \cdot x^4)) \)

2. \( 2 \cdot x^{2-1} + \frac{d}{dx} \sin(3 \cdot x^4) \)

3. \( 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \)

How can we guide the student to the next formula (no. 4.)

\[ \frac{d}{dx}(x^2 + \sin(3 \cdot x^4)) \]

... or some algebraically equivalent formula?

4. \( 2 \cdot x + \cos(3 \cdot x^4) \cdot \frac{d}{dx}(3 \cdot x^4) \)
Dialog mechanically generated by CTP-technology

1. \( \frac{d}{dx} (x^2 + \sin(3 \cdot x^4)) \)
2. \( 2 \cdot x^{2-1} + \frac{d}{dx} \sin(3 \cdot x^4) \)
3. \( 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \)

4. ________________
Dialog mechanically generated by CTP-technology

\[ 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \]

4. 2__________________
Dialog mechanically generated by CTP-technology

3. \[ 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \]

4. \[ 2 \cdot \underline{_________________} \]
Dialog mechanically generated by CTP-technology

\[ 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \]

\[ 2 \cdot x \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}\]
Dialog mechanically generated by CTP-technology

3. \(2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4)\)

4. \(2 \cdot x + \underline{\text{___________}}\)
Dialog mechanically generated by CTP-technology

\[ 3. \quad 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \]

\[ 4. \quad 2 \cdot x + \cos(3 \cdot x^4) \]

Input checked by a prover.
Dialog mechanically generated by CTP-technology

\[
\begin{align*}
3. \quad 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \\
\frac{d}{dx} \sin(u) &= \cos(u) \cdot \frac{d}{dx}\
\end{align*}
\]

Provers identify and suggest theorems.
Dialog mechanically generated by CTP-technology

\[
3. \quad 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4)
\]

\[
\frac{d}{dx} \sin(u) = \cos(u) \cdot \frac{d}{dx} \, ???
\]
Dialog mechanically generated by CTP-technology

3. \[ 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \]

\[ \frac{d}{dx} \cos(x) = -\sin(x) \]
\[ \frac{d}{dx} \sin(u) = \cos(u) \cdot \frac{d}{dx} u \]
\[ \frac{d}{dx} x^n = n \cdot x^{n-1} \]

Provers operate on theories comprising theorems.
Dialog mechanically generated by CTP-technology

\[
3. \quad 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4)
\]
\[
\frac{d}{dx} \cos(x) = -\sin(x)
\]
\[
\frac{d}{dx} \sin(u) = \cos(u) \cdot \frac{d}{dx} u
\]
\[
\frac{d}{dx} x^n = n \cdot x^{n-1}
\]
Dialog mechanically generated by CTP-technology

3. \[ 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \]

\[ \frac{d}{dx} \sin(u) = \cos(u) \cdot \frac{d}{dx} u \]

4. \[ 2 \cdot x + \cos(3 \cdot x^4) \cdot \frac{d}{dx} \__ \]

Provers use “matching” for fill-in gaps.
Dialog mechanically generated by CTP-technology

\[
\begin{align*}
3. & \quad 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \\
4. & \quad 2 \cdot x + \cos(3 \cdot x^4) \cdot \frac{d}{dx}\left(\text{_____}\right)
\end{align*}
\]
Dialog mechanically generated by CTP-technology

3. \[ 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \]

\[
\frac{d}{dx} \sin(u) = \cos(u) \cdot \frac{d}{dx} u
\]

4. \[ 2 \cdot x + \cos(3 \cdot x^4) \cdot \frac{d}{dx}(3____) \]
Dialog mechanically generated by CTP-technology

3. \[ 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \]

\[ \frac{d}{dx} \sin(u) = \cos(u) \cdot \frac{d}{dx} u \]

4. \[ 2 \cdot x + \cos(3 \cdot x^4) \cdot \frac{d}{dx} (3 \cdot \underline{\hspace{1cm}}) \]
Dialog mechanically generated by CTP-technology

\[ \begin{align*}
3. & \quad 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \\
\frac{d}{dx} \sin(u) &= \cos(u) \cdot \frac{d}{dx} u \\
4. & \quad 2 \cdot x + \cos(3 \cdot x^4) \cdot \frac{d}{dx} (3 \cdot x) 
\end{align*} \]
Dialog mechanically generated by CTP-technology

\[ 3. \quad 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \]

\[ \frac{d}{dx} \sin(u) = \cos(u) \cdot \frac{d}{dx} u \]

\[ 4. \quad 2 \cdot x + \cos(3 \cdot x^4) \cdot \frac{d}{dx} (3 \cdot x^4) \_ \]
Dialog mechanically generated
by CTP-technology

\[ 3. \quad 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \]

\[ \frac{d}{dx} \sin(u) = \cos(u) \cdot \frac{d}{dx} u \]

\[ 4. \quad 2 \cdot x + \cos(3 \cdot x^4) \cdot \frac{d}{dx} (3 \cdot x^4) \]
Dialog mechanically generated by CTP-technology

3.   \[ 2 \cdot x + \frac{d}{dx} \sin(3 \cdot x^4) \]

\[ \frac{d}{dx} \sin(u) = \cos(u) \cdot \frac{d}{dx} u \]

4.   \[ 2 \cdot x + \cos(3 \cdot x^4) \cdot \frac{d}{dx} (3 \cdot x^4) \]

Provers check, if a formula can be derived in a context.
Requirements and Technologies

The requirements can be met by the technologies

1. **check user input** as generous and liberal as possible
   - by Computer Theorem Proving (CTP), i.e.
   - Automated Proving (ATP): simplifiers, SAT, SMT etc
   - Interactive Proving (ITP): Coq and Isabelle in EU

2. **guide the user** step by step towards a solution
   - by *interpretation in debug-mode* with breakpoints
   - ? *resume* after liberal user input at breakpoint (ATP !)

3. **explain steps** on request by the user
Outline

1. Remarks on tutoring software
   Status quo in the design
   Requirements on the design

2. Base technologies for tutoring
   Computer theorem proving
   Single-stepping program interpretation
   Human readable math knowledge

3. Summary and Conclusion
Requirements and Technologies

The requirements can be met by the technologies

1. **check user input** as generous and liberal as possible
   - by Computer Theorem Proving (CTP), i.e.
   - Automated Proving (ATP): simplifiers, SAT, SMT etc
   - Interactive Proving (ITP): Coq and Isabelle in EU

2. **guide the user** step by step towards a solution
   - by interpretation in debug-mode with breakpoints
   - ? resume after liberal user input at breakpoint (ATP !)

3. **explain steps** on request by the user
   - by human-readable knowledge on a separate language layer.
The requirements can be met by the technologies

1. **check user input** as generous and liberal as possible
   - by Computer Theorem Proving (CTP), i.e.
   - Automated Proving (ATP): simplifiers, SAT, SMT etc
   - Interactive Proving (ITP): Coq and Isabelle in EU

2. **guide the user** step by step towards a solution
   - by interpretation in debug-mode with breakpoints
   - ? resume after liberal user input at breakpoint (ATP !)

3. **explain steps** on request by the user
   - by **human-readable knowledge** on a separate language layer.
Demo Isabelle distribution
Summary

General features and technologies like . . .

1. **check user input** — comp. theorem proving (CTP)
2. **guide the user** — interpretation in debug-mode
3. **explain steps** — human-readable knowledge

. . . provide for novel services in tutoring and authoring.

Austria can contribute significantly!
Summary

General features and technologies like . . .

1. check user input — comp. theorem proving (CTP)
2. guide the user — interpretation in debug-mode
3. explain steps — human-readable knowledge

. . . provide for novel services in tutoring and authoring.

Austria can contribute significantly!
Summary

General features and technologies like …

1. check user input — comp. theorem proving (CTP)
2. guide the user — interpretation in debug-mode
3. explain steps — human-readable knowledge

…provide for novel services in tutoring and authoring.

Austria can contribute significantly!
Conclusion?

Possible Austrian contributions are . . .

- **interpretation in debug-mode**, \textit{TSAC} + Isabelle
  - TU Graz, Institute for Softwaretechnology

- **user-guidance** in single-stepping
  - TU Graz, IICM (H. Maurer)

- **Computer Algebra**
  - RISC Linz (B. Buchberger)

- **Dynamic Geometry**
  - Uni Linz (GeoGebra, M. Hohenwarter)

- **Didactics, teacher training, field tests**
  - ACDCA

Joint development of partners above?
Start with a course by Ralph-Johan Back?
Conclusion?

Possible Austrian contributions are . . .

- . . . interpretation in debug-mode, \texttt{TSAC}+\texttt{Isabelle}
  - TU Graz, Institute for Softwaretechnology

- . . . user-guidance in single-stepping
  - TU Graz, IICM (H.Maurer)

- . . . Computer Algebra
  - RISC Linz (B.Buchberger)

- . . . Dynamic Geometry
  - Uni Linz (GeoGebra, M.Hohenwarter)

- . . . Didactics, teacher training, field tests
  - ACDCA

Joint development of partners above?
Start with a course by Ralph-Johan Back?