An Isabelle-based educational math assistant
present and future development
How continue cooperation?

Walther Neuper

Institute for Softwaretechnology
Graz University of Technology

Meeting with the Isabelle Developer Team
February 23, 2011
TU Munich
State of development in the ISAC-project
User requirements from experiments at high schools
Demo of the experimental ISAC-system
Lucas-Interpreter combines computation and deduction

ISAC’s architecture and future development
Survey on the plans
Details about the plans
What is a document model?
? Project: CTP-based programming language

Summary
1. State of development in the ISAC-project
   - User requirements from experiments at high schools
   - Demo of the experimental ISAC-system
   - Lucas-Interpreter combines computation and deduction

2. ISAC’s architecture and future development
   - Survey on the plans
   - Details about the plans
   - What is a document model?
   - Project: CTP-based programming language

3. Summary
The ideal math assistant . . .

1. guides the user step by step towards a solution

   Watching teachers calculate step by step is boring. Operating on formulas by hand is hard, too. Software can support independent learning.

2. checks user input as generous and liberal as possible

   Active learning by trial and error ist most effective. Programmers cannot foresee learners’ input. Theorem provers provide most general checking.

3. explains steps on request by the user

   Programmers also cannot foresee learners’ questions. A system must be transparent for casual questions. LCF-style provers have knowledge human readable.
The ideal math assistant . . .

1 guides the user step by step towards a solution  
   Watching teachers calculate step by step is boring.  
   Operating on formulas by hand is hard, too.  
   Software can support **independent learning**.

2 checks user input as generous and liberal as possible  
   Active learning by **trial and error** is most effective.  
   Programmers cannot foresee learners’ input.  
   Theorem provers provide most general checking.

3 explains steps on request by the user  
   Programmers also cannot foresee learners’ questions.  
   A system must be **transparent** for casual questions.  
   LCF-style provers have knowledge human readable.
The ideal math assistant . . .

1 guides the user step by step towards a solution
   Watching teachers calculate step by step is boring. Operating on formulas by hand is hard, too. Software can support independent learning.
2 checks user input as generous and liberal as possible
   Active learning by trial and error ist most effective. Programmers cannot foresee learners’ input. Theorem provers provide most general checking.
3 explains steps on request by the user
   Programmers also cannot foresee learners’ questions. A system must be transparent for casual questions. LCF-style provers have knowledge human readable.
The ideal math assistant . . .

1. **guides the user** step by step towards a solution
   
   Watching teachers calculate step by step is boring.
   Operating on formulas by hand is hard, too.
   **Software can support independent learning.**

2. **checks user input** as generous and liberal as possible
   
   Active learning by **trial and error** is most effective.
   Programmers cannot foresee learners’ input.
   **Theorem provers provide most general checking.**

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

   Programmers also cannot foresee learners’ questions.
   **A system must be transparent for casual questions.**
**LCF-style provers have knowledge human readable.**
1. **State of development in the ISAC-project**
   - User requirements from experiments at high schools
   - Demo of the experimental ISAC-system
   - Lucas-Interpreter combines *computation* and *deduction*

2. **ISAC's architecture and future development**
   - Survey on the plans
   - Details about the plans
   - What is a document model?
   - Project: CTP-based programming language

3. **Summary**
Demo of the $\text{ISAC}$-system

1. passive Webpages
   ((interactive pages are broken due to ongoing update from Java1.5 to Java1.6))

2. interactive system
1. State of development in the ISAC-project
   User requirements from experiments at high schools
   Demo of the experimental ISAC-system
   Lucas-Interpreter combines computation and deduction

2. ISAC’s architecture and future development
   Survey on the plans
   Details about the plans
   What is a document model?
   ? Project: CTP-based programming language

3. Summary
Lucas-interpretation combines *proving* and *programming*
The ISAC-prototype is based on Isabelle

Isabelle/Isar
logical operating system
(contexts etc)

Interpretation

Specification

Program

Proving and Programming
Programs usually produce output.

Isabelle/Isar
logical operating system
(contexts etc)

specification

program

output

Production
Lucas-Interpreter is a debugger in single-stepping mode.
Given a specification and a program . . .
... tutoring starts with precondition fulfilled

Tutoring

_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
 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
Breakpoint 1: user accepts or updates or inputs

Isabelle — ISAC
Walther Neuper

State
Requirements
Demo ISAC
Lucas-Interpreter

Plans
Survey
Plans in detail
Document model
Project

Summary

worksheet
Problem (B, bendl)
Problem (B, load2bl)
Q(x) = c-q.x, M(x) = ...

dialog module

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

Tutoring
Breakpoint 2: user accepts or updates or inputs

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...

Dialog module

Tutoring

Isabelle/Isar
Logical operating system (contexts etc)

Interpreter (PLI)

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
Breakpoint 3: user accepts *or* updates *or* inputs

Tutoring
Breakpoint 4: user accepts or updates or inputs

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…

Isabelle/Isar
logical operating system
(contexts etc)

Spec:
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

program

output

Tutoring
Breakpoint 5: user accepts or updates or inputs

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 …

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

Tutoring
Problem solved with postcondition fulfilled

worksheet

Problem (B, bendl)
Problem (B, load2bl)
\( Q(x) = c \cdot q \cdot x \), \( M(x) = \ldots \)
Problem (B, sidecds)
\( L \cdot q = x \), \( 0 = c_2 + L \cdot c \ldots \)
solveSys \( [0=c_3, \ldots \)
c \( = q \cdot L \), \( c_2 = -L^2 \cdot q / 2 \)
y\( (x) = c_4 + c_3 \cdot x - 1 / E I \ldots \)
y\( (x) = 0 + 0 \cdot x - 1 / E I \ldots \)
y\( (x) = (q \cdot L^2) / (4 \cdot E I) \cdot x^2 - (q \cdot L) / (6 \cdot E I) \cdot x^3 + q / (24 \cdot E I) \cdot x^4 \)

Tutoring

dialog module

Isabelle/Isar
logical operating system
(contexts etc)

interpreter (P.L)

specification

Program (B, bendl)
Program (B, load2bl)
\( Q(x) = c \cdot q \cdot x \), \( M(x) = \ldots \)
Program (B, sidecds)
\( L \cdot q = x \), \( 0 = c_2 + L \cdot c \ldots \)
solveSys \( [0=c_3, \ldots \)
c \( = q \cdot L \), \( c_2 = -L^2 \cdot q / 2 \)
y\( (x) = c_4 + c_3 \cdot x - 1 / E I \ldots \)
y\( (x) = 0 + 0 \cdot x - 1 / E I \ldots \)
y\( (x) = (q \cdot L^2) / (4 \cdot E I) \cdot x^2 - (q \cdot L) / (6 \cdot E I) \cdot x^3 + q / (24 \cdot E I) \cdot x^4 \)

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
Specification for 'Biegenlinie'

\[\text{in} : \text{function } q_0, \text{length } L\]
\[\text{pre} : \quad q_0 \text{ is integrable in } x \land L > 0\]
\[\text{out} : \quad \text{function } y(x)\]
\[\text{post} : \quad y(0) = 0 \land y'(0) = 0 \land V(0) = q_0 \cdot L \land M_b(L) = 0\]

where \( V \) and \( M_b \) are constant function symbols in the theory of “bending lines”. \text{function} and \text{length} are functions fixing the arguments’ types; \( q_0 \) is a constant function with type \( \mathcal{R} \to \mathcal{R} \).
Program for 'Biegelinie'

01 Script bendingLine
02 (l_::real) (q_::real) (v_::real) (b_::real=:real) (rb_::bool list) =
03 (LET
04 (funs_: bool list) =
05 (SubProblem (Bendingline,[bendingline,integrate],
06 [bendingline,integrate])
07 [real_ q_, real_real_ b_, real_ v_]);
08 (equus_:bool list) =
09 (SubProblem (Bendingline,[bendingline,setConstraints],
10 [bendingline,setConstraints])
11 [bools_ funs_, bools_ rb_, real_ l_]);
12 (sols_:bool list) =
13 (SubProblem (Real,[equation,system,linear],[])
14 [bools_ equus_, reals_ [c,c_2,c_3,c_4]]);
15 B_ = Take (LAST funs_);
16 B_ = ((Substitute sols_) @@
17 (Rewrite_Set_Inst [(bdv, v_) make_ratpoly_in)) B_
18 IN B_)
These are breakpoints:

01 Script bendingLine
02 (l_::real) (q_::real) (v_::real) (b_::real=>real) (rb_::bool list) =
03 (LET
04 (funs_:: bool list) =
05 (SubProblem (Bendingline,[bendingline,integrate],
06 [bendingline,integrate])
07 [real_ q_, real_real_ b_, real_ v_]);
08 (equus_::bool list) =
09 (SubProblem (Bendingline,[bendingline,setConstraints],
10 [bendingline,setConstraints])
11 [bools_ funs_, bools_ rb_, real_ l_]);
12 (sols_::bool list) =
13 (SubProblem (Real,[equation,system,linear],[])
14 [bools_ equus_, reals_ [c,c_2,c_3,c_4]]);
15 B_ = Take (LAST funs_);
16 B_ = ((Substitute sols_) @@
17 (Rewrite_Set_Inst [(bdv, v_)] make_ratpoly_in)) B_
18 IN B_)
Contexts by Lucas-Interpreter

01..03 *Script bendingLine . . .*

\[
ctxt_0 = \{(l, L), (q, q_0), (v, x), (b, y), (rb, [y(0) = 0, y′(0) = 0, V(0) = q_0 \cdot L, 
q_0 \text{ is integrable in } x, \ L > 0]}\]

04..07 *(funs::_:: bool list) = . . .

\[
ctxt_1 = ctxt_0 \cup \{(\text{funs}_\_,[y(x) = \frac{q_0 \cdot L^2}{4 \cdot EI} \cdot x^2 - \frac{q_0 \cdot L}{6 \cdot EI} \cdot x^3 + \frac{q_0}{24 \cdot EI} \cdot x^4, 
\ y'(x) = \frac{q_0 \cdot L^2}{2 \cdot EI} \cdot x - \frac{q_0 \cdot L}{2 \cdot EI} \cdot x^2 + \frac{q_0}{6 \cdot EI} \cdot x^3, . . . ])}\}

08..11 *(equus::_::bool list) = . . .

\[
ctxt_2 = ctxt_1 \cup \{\}
\]

12..15 *(sols::_::bool list) = . . .

\[
ctxt_3 = ctxt_2 \cup \{\}
\]

15 \[
B_ = \text{Take (LAST funs_)};
\]

\[
ctxt_4 = ctxt_3 \cup \{\}
\]

16 \[
B_ = ((\text{Substitute sols_}) @@
\]

\[
ctxt_5 = ctxt_4 \cup \{\}
\]

17 \[
(Rewrite_Set_Inst [(bdv, v_)] \text{ make_ratpoly_in})) B_\]

\[
ctxt_6 = ctxt_5 \cup \{\}
\]

18 \[
\text{IN } B_\)
Contexts by Lucas-Interpreter

01..03  *Script bendingLine* . . .
\[ctxt_0 = \{(l_-, L), (q_-, q_0), (v_-, x), (b_-, y), (rb, [y(0) = 0, y'(0) = 0, V(0) = q_0 \cdot L, q_0 \text{ is integrable in } x, L > 0]}\]

04..07  \((\text{funs}:: \text{bool list}) = . . .\)
\[ctxt_1 = ctxt_0 \cup \{((\text{funs}_-, [y(x) = \frac{q_0 \cdot L^2}{4 \cdot EI} \cdot x^2 - \frac{q_0 \cdot L}{2 \cdot EI} \cdot x^3 + \frac{q_0}{24 \cdot EI} \cdot x^4, y'(x) = \frac{q_0 \cdot L^2}{2 \cdot EI} \cdot x - \frac{q_0 \cdot L}{2 \cdot EI} \cdot x^2 + \frac{q_0}{6 \cdot EI} \cdot x^3, \ldots])\}\]

08..11  \((\text{equs}::\text{bool list}) = . . .\)
\[ctxt_2 = ctxt_1 \cup \{}
\]

12..15  \((\text{sols}::\text{bool list}) = . . .\)
\[ctxt_3 = ctxt_2 \cup \{}
\]

15  \[B_ = \text{Take} (\text{LAST} \text{ funs}_-);\]
\[ctxt_4 = ctxt_3 \cup \{}
\]

16  \[B_ = ((\text{Substitute} \text{ sols}_-) @@\]
\[ctxt_5 = ctxt_4 \cup \{}
\]

17  \[\text{IN} B_\]
\[ctxt_6 = ctxt_5 \cup \{}
\]

18  \[\text{IN} B_\]
01..03 Script bendingLine . . .
ctxt_0 = \{(l_, L), (q_, q_0), (v_, x), (b_, y), (rb, [y(0) = 0, y'(0) = 0, V(0) = q_0 \cdot L, q_0 is integrable in x, L > 0]}
04..07 \{(funs_:: bool list) = . . .
ctxt_1 = ctxt_0 \cup \{(funs_, [y(x) = \frac{q_0 \cdot L^2}{4 \cdot EI} \cdot x^2 - \frac{q_0 \cdot L}{6 \cdot EI} \cdot x^3 + \frac{q_0}{24 \cdot EI} \cdot x^4, y'(x) = \frac{q_0 \cdot L^2}{2 \cdot EI} \cdot x - \frac{q_0 \cdot L}{2 \cdot EI} \cdot x^2 + \frac{q_0}{6 \cdot EI} \cdot x^3, \ldots])\}
08..11 \{(equus_::bool list) = . . .
cxtt_2 = ctxt_1 \cup \{}
12..15 \{(sols_::bool list) = . . .
cxtt_3 = ctxt_2 \cup \{}
15 B_ = Take (LAST funs_);
cxtt_4 = ctxt_3 \cup \{}
16 B_ = ((Substitute sols_) @@
cxtt_5 = ctxt_4 \cup \{}
17 (Rewrite_Set_Inst [(bdv, v_)] make_ratpoly_in)) B_
cxtt_6 = ctxt_5 \cup \{}
18 IN B_\)
Contexts by Lucas-Interpreter

01..03 Script bendingLine . . .
ctxt₀ = \{(l, L), (q, q₀), (v, x), (b, y), (rb, [y(0) = 0, y′(0) = 0, V(0) = q₀·L]),
q₀ is_integrable_in x, L > 0\}

04..07 (funs_:: bool list) = . . .
ctxt₁ = ctxt₀ ∪ \{(funs_, [y(x) = \frac{q₀·L²}{4·EI} · x² - \frac{q₀·L}{6·EI} · x³ + \frac{q₀}{24·EI} · x⁴,
y′(x) = \frac{q₀·L²}{2·EI} · x - \frac{q₀·L}{2·EI} · x² + \frac{q₀}{6·EI} · x³, . . .])\}

08..11 (equ_::bool list) = . . .
ctxt₂ = ctxt₁ ∪ {}

12..15 (sols_::bool list) = . . .
ctxt₃ = ctxt₂ ∪ {}

15 B_ = Take (LAST funs_);
ctxt₄ = ctxt₃ ∪ {}  

16 B_ = ((Substitute sols_) @@
ctxt₅ = ctxt₄ ∪ {}  

17 (Rewrite_Set_Inst [(bdv, v_)] make_ratpoly_in)) B_
ctxt₆ = ctxt₅ ∪ {}  

18 IN B_)
Contexts by Lucas-Interpreter

01..03 Script bendingLine . . .
ctxt₀ = {(l_, L), (q_, q₀), (v_, x), (b_, y), (rb, [y(0) = 0, y′(0) = 0, V(0) = q₀.L]),
q₀ is_integrable_in x, L > 0}

04..07 (funs_::bool list) = . . .
ctxt₁ = ctxt₀ ∪ {(funs_, [y(x) = \frac{q₀.L²}{4.EI} \cdot x² - \frac{q₀.L}{6.EI} \cdot x³ + \frac{q₀}{24.EI} \cdot x⁴, y′(x) = \frac{q₀.L²}{2.EI} \cdot x - \frac{q₀.L}{2.EI} \cdot x² + \frac{q₀}{6.EI} \cdot x³, . . .])}

08..11 (equus_::bool list) = . . .
ctxt₂ = ctxt₁ ∪ {}

12..15 (sols_::bool list) = . . .
ctxt₃ = ctxt₂ ∪ {}

15 B_ = Take (LAST funs_);
ctxt₄ = ctxt₃ ∪ {}

16 B_ = ((Substitute sols_) @@
ctxt₅ = ctxt₄ ∪ {}

17 (Rewrite_Set_Inst [(bdv, v_)] make_ratpoly_in)) B_
ctxt₆ = ctxt₅ ∪ {}

18 IN B_
Contexts by Lucas-Interpreter

01..03 Script bendingLine . .
ctxt\_0 = \{(l\_\_, L), (q\_\_, q\_0), (v\_\_, x), (b\_\_, y), (rb, [y(0) = 0, y'(0) = 0, V(0) = q\_0 \cdot L, q\_0 \text{ is integrable in } x, L > 0]}

04..07 (funs\_:: bool list) = . .
ctxt\_1 = ctxt\_0 \cup \{(funs\_, [y(x) = \frac{q\_0 \cdot L^2}{4 \cdot EI} \cdot x^2 - \frac{q\_0 \cdot L}{6 \cdot EI} \cdot x^3 + \frac{q\_0}{24 \cdot EI} \cdot x^4, y'(x) = \frac{q\_0 \cdot L^2}{2 \cdot EI} \cdot x - \frac{q\_0 \cdot L}{2 \cdot EI} \cdot x^2 + \frac{q\_0}{6 \cdot EI} \cdot x^3, \ldots])\}

08..11 (equus\_::bool list) = . .
ctxt\_2 = ctxt\_1 \cup \{}

12..15 (sols\_::bool list) = . .
ctxt\_3 = ctxt\_2 \cup \{}

15 B_ = Take (LAST funs\_);
ctxt\_4 = ctxt\_3 \cup \{}

16 B_ = ((Substitute sols\_) @@
ctxt\_5 = ctxt\_4 \cup \{}

17 (Rewrite_Set_Inst [(bdv, v\_)] make_ratpoly_in)) B_
ctxt\_6 = ctxt\_5 \cup \{}

18 IN B_

Contexts by Lucas-Interpreter

01..03 *Script bendingLine* . . .
\[
ctxt_0 = \{(l_, L), (q_, q_0), (v_, x), (b_, y), (rb, [y(0) = 0, y'(0) = 0, V(0) = q_0 \cdot L])
\]
\[
\text{\textit{q}_0 \text{ is}\_\text{integrable\_in} x, L > 0}\}
\]

04..07 \( (\text{funs\_:: bool list}) = \ldots \)
\[
ctxt_1 = ctxt_0 \cup \{(\text{funs\_}, [y(x) = \frac{q_0 \cdot L^2}{4 \cdot EI} \cdot x^2 - \frac{q_0 \cdot L}{6 \cdot EI} \cdot x^3 + \frac{q_0}{24 \cdot EI} \cdot x^4, y'(x) = \frac{q_0 \cdot L^2}{2 \cdot EI} \cdot x - \frac{q_0 \cdot L}{2 \cdot EI} \cdot x^2 + \frac{q_0}{6 \cdot EI} \cdot x^3, \ldots])\}\}
\]

08..11 \( (\text{equus\_::bool list}) = \ldots \)
\[
cctxt_2 = ctxt_1 \cup \{\}
\]

12..15 \( (\text{sols\_::bool list}) = \ldots \)
\[
cctxt_3 = ctxt_2 \cup \{\}
\]

15 \( B_\_ = \text{Take (LAST funs\_)}; \)
\[
cctxt_4 = ctxt_3 \cup \{\}
\]

16 \( B_\_ = ((\text{Substitute sols\_}) @@ \)
\[
cctxt_5 = ctxt_4 \cup \{\}
\]

17 \( (\text{Rewrite\_Set\_Inst [(bdv, v\_)] make\_ratpoly\_in}) B_\)\)
\[
cctxt_6 = ctxt_5 \cup \{\}
\]

18 \( \text{IN B_}\)
Contexts by Lucas-Interpreter

01..03 *Script bendingLine* . . .

\[ \text{ctxt}_0 = \{(l_, L), (q_, q_0), (v_, x), (b_, y), (rb, [y(0) = 0, y'(0) = 0, V(0) = q_0 \cdot L])} \]

q_0 \text{ is integrable in } x, \ L > 0

04..07 \( \text{(funs_::bool list)} = \ldots \)

\[ \text{ctxt}_1 = \text{ctxt}_0 \cup \{(\text{funs}_-, [y(x) = \frac{q_0 \cdot L^2}{4 \cdot EI} \cdot x^2 - \frac{q_0 \cdot L}{6 \cdot EI} \cdot x^3 + \frac{q_0}{24 \cdot EI} \cdot x^4,}
\]
\[ y'(x) = \frac{q_0 \cdot L^2}{2 \cdot EI} \cdot x - \frac{q_0 \cdot L}{2 \cdot EI} \cdot x^2 + \frac{q_0}{6 \cdot EI} \cdot x^3, \ldots ] \}\}

08..11 \( \text{(equus_::bool list)} = \ldots \)

\[ \text{ctxt}_2 = \text{ctxt}_1 \cup \{\} \]

12..15 \( \text{(sols_::bool list)} = \ldots \)

\[ \text{ctxt}_3 = \text{ctxt}_2 \cup \{\} \]

15 \[ B_ = \text{Take (LAST funs_);} \]

\[ \text{ctxt}_4 = \text{ctxt}_3 \cup \{\} \]

16 \[ B_ = ((\text{Substitute sols_}) @@ \]

\[ \text{ctxt}_5 = \text{ctxt}_4 \cup \{\} \]

17 \( \text{(Rewrite_Set_Inst [(bdv, v_) make_ratpoly_in]) B_} \)

\[ \text{ctxt}_6 = \text{ctxt}_5 \cup \{\} \]

18 \( \text{IN B_} \)
Lucas-Interpretation also in geometry
Somewhere during stepwise construction . . .

context =
\[ A \neq B, B \neq C, C \neq A, \text{--collinear } A B C \]

\[ BM_1 \parallel BC, \quad \frac{BM_1}{BC} = \frac{1}{2}, \quad M_2 N_2 \perp BC, \quad \frac{4S_{M_2BN_2}}{P_{M_2BM_2}} = 1 \]
...the user inputs the next step: logical data created!
Lucas-Interpretation compares with subsequent contexts

context =
A ≠ B, B ≠ C, C ≠ A, \( -\text{collinear } A B C \)

\[ \frac{BM_1}{BC} = 1/2, M_2N_3 \perp BC, \frac{4S_{M_1BN_3}}{P_{M_1BM_3}} = 1 \]

\[ \frac{BM_3}{BA} = 1/2, M_3N_3 \perp BA, \frac{4S_{M_3BN_3}}{P_{M_3BM_3}} = 1 \]

\[ \frac{AM_2}{AC} = 1/2, M_2N_2 \perp AC, \frac{4S_{M_2AN_2}}{P_{M_2AM_2}} = 1 \]
Lucas-Interpretation compares with subsequent contexts

Isabelle/Isar

Lucas-Interpreter

context =

\[ A \neq B, B \neq C, C \neq A, \neg \text{collinear } A B C \]
\[ BM_1 \parallel BC, \frac{BM_1}{BC} = 1/2, M_1N_1 \perp BC, \frac{4S_{M_1BN_1}}{P_{M_1BM_1}} = 1 \]
\[ BM_3 \parallel BA, \frac{BM_3}{BA} = 1/2, M_3N_3 \perp BA, \frac{4S_{M_3BN_3}}{P_{M_3BM_3}} = 1 \]
\[ AM_2 \parallel AC, \frac{AM_2}{AC} = 1/2, M_2N_2 \perp AC, \frac{4S_{M_2AN_2}}{P_{M_2AM_2}} = 1 \]

\[ \text{collinear } N_1 M_1 O, \text{ collinear } N_2 M_2 O \]
The experiments show the system’s feasibility to...

1. **guide the user** step by step towards a solution.
   ...step from **breakpoint to breakpoint** in a program.
   The user accepts *or* updates *or* inputs (dialog module!)

2. **check user input** as generously.
   ...provide provers with **logical context** of statements.
   Checking user-input is: prove derivability from context.

3. **explain steps** on request by the user.
   ...interprete **human-readable** knowledge of Isabelle.
   Knowledge shall be interlinked with a mathematics wiki.

...this is called **Lucas-Interpretation**
The experiments show the system’s feasibility to . . .

1. guide the user step by step towards a solution. . . step from **breakpoint to breakpoint** in a program. The user accepts *or* updates *or* inputs (dialog module!)

2. check user input as generously. . . provide provers with **logical context** of statements. Checking user-input is: prove derivability from context.

3. explain steps on request by the user. . . interprete **human-readable** knowledge of Isabelle. Knowledge shall be interlinked with a mathematics wiki.

...this is called **Lucas-Interpretation**
The experiments show the system’s feasibility to . . .

1. **guide the user** step by step towards a solution.
   . . . step from **breakpoint to breakpoint** in a program. The user accepts *or* updates *or* inputs (dialog module!)

2. **check user input** as generously.
   . . . provide provers with **logical context** of statements. Checking user-input is: prove derivability from context.

3. **explain steps** on request by the user.
   . . . interprete **human-readable** knowledge of Isabelle. Knowledge shall be interlinked with a mathematics wiki.

...this is called **Lucas-Interpretation**
The experiments show the system’s feasibility to . . .

1. **guide the user** step by step towards a solution.
   . . . step from **breakpoint to breakpoint** in a program.
   The user accepts *or* updates *or* inputs (dialog module!)

2. **check user input** as generously.
   . . . provide provers with **logical context** of statements.
   Checking user-input is: prove derivability from context.

3. **explain steps** on request by the user.
   . . . interprete **human-readable** knowledge of Isabelle.
   Knowledge shall be interlinked with a mathematics wiki.

...this is called **Lucas-Interpretation**
The experiments show the system’s feasibility to . . .

1. **guide the user** step by step towards a solution.
   . . . step from **breakpoint to breakpoint** in a program. The user accepts *or* updates *or* inputs (dialog module!)

2. **check user input** as generously.
   . . . provide provers with **logical context** of statements. Checking user-input is: prove derivability from context.

3. **explain steps** on request by the user.
   . . . interprete **human-readable** knowledge of Isabelle. Knowledge shall be interlinked with a mathematics wiki.

. . . this is called **Lucas-Interpretation**
The experiments show the system’s feasibility to . . .

1. **guide the user** step by step towards a solution.
   . . . step from **breakpoint to breakpoint** in a program.
   The user accepts *or* updates *or* inputs (dialog module!)

2. **check user input** as generously.
   . . . provide provers with **logical context** of statements.
   Checking user-input is: prove derivability from context.

3. **explain steps** on request by the user.
   . . . interprete **human-readable** knowledge of Isabelle.
   Knowledge shall be interlinkd with a mathematics wiki.

...this is called **Lucas-Interpretation**
The experiments show the system’s feasibility to . . .

1. **guide the user** step by step towards a solution. . . step from **breakpoint to breakpoint** in a program. The user accepts *or* updates *or* inputs (dialog module!)

2. **check user input** as generously. . . provide provers with **logical context** of statements. Checking user-input is: prove derivability from context.

3. **explain steps** on request by the user. . . interprete **human-readable** knowledge of Isabelle. Knowledge shall be interlinked with a mathematics wiki.

...this is called **Lucas-Interpretation**
The experiments show the system’s feasibility to . . .

1. **guide the user** step by step towards a solution. . . step from **breakpoint to breakpoint** in a program. The user accepts *or* updates *or* inputs (dialog module!)

2. **check user input** as generously. . . provide provers with **logical context** of statements. Checking user-input is: prove derivability from context.

3. **explain steps** on request by the user. . . interprete **human-readable** knowledge of Isabelle. Knowledge shall be interlinked with a mathematics wiki.

. . . this is called **Lucas-Interpretation**
1 State of development in the ISAC-project
   User requirements from experiments at high schools
   Demo of the experimental ISAC-system
   Lucas-Interpreter combines computation and deduction

2 ISAC’s architecture and future development
   Survey on the plans
   Details about the plans
   What is a document model?
   ? Project: CTP-based programming language

3 Summary
**ISAC**’s present experimental version
Version under construction
(Isars contexts)
Next version: generalized front-end
Could there be Standard ML instead the Isabelle terms?
State of development in the *ISAC*-project
User requirements from experiments at high schools
Demo of the experimental *ISAC*-system
Lucas-Interpreter combines *computation* and *deduction*

*ISAC’s* architecture and future development
Survey on the plans
Details about the plans
What is a document model?
? Project: CTP-based programming language

Summary
See the blackboard!
Outline

1. State of development in the ISAC-project
   User requirements from experiments at high schools
   Demo of the experimental ISAC-system
   Lucas-Interpreter combines \textit{computation} and \textit{deduction}

2. ISAC’s architecture and future development
   Survey on the plans
   Details about the plans
   \textbf{What is a document model ?}
   ? Project: CTP-based programming language

3. Summary
What is a document model?
Participants in learning?
1. State of development in the \textit{ISAC}-project
   
   User requirements from experiments at high schools
   
   Demo of the experimental \textit{ISAC}-system
   
   Lucas-Interpreter combines \textit{computation} and \textit{deduction}

2. \textit{ISAC}'s architecture and future development
   
   Survey on the plans
   
   Details about the plans
   
   What is a document model? 
   
   ? Project: CTP-based programming language

3. Summary
CTP-based programming language

• **We know**: CAS\(^1\)-based programming languages are
  • the most used in science and technology
  • not reliable by design (dop solutions etc)

• **We plan**: a CTP\(^2\)-based programming language, which
  • is based on Isabelle's (growing!) math knowledge
  • is reliable because based on logic
  • uses Isar/Scala/jEdit as a frontend
  • extends jEdit (JavaSwing!) with a formula editor

• **Potential partners**
  • Isabelle at TU Munich
  • Research Inst. for Symbolic Computation (RISC), Linz
  • Inst. of Computer Languages TU Vienna
  • Inst. for Softwaretechnology TU Graz

\(^1\)CAS = Computer Algebra System
\(^2\)CTP = Computer Theorem Prover
CTP-based programming language

- We know: CAS\(^1\)-based programming languages are
  - the most used in science and technology
  - not reliable by design (dop solutions etc)
- We plan: a CTP\(^2\)-based programming language, which
  - is based on Isabelle’s (growing!) math knowledge
  - is reliable because based on logic
  - uses Isar/Scala/jEdit as a frontend
  - extends jEdit (JavaSwing!) with a formula editor

Potential partners
- Isabelle at TU Munich
- Research Inst. for Symbolic Computation (RISC), Linz
- Inst. of Computer Languages TU Vienna
- Inst. for Softwaretechnology TU Graz

\(^1\)CAS = Computer Algebra System
\(^2\)CTP = Computer Theorem Prover
CTP-based programming language

- We know: CAS\(^1\)-based programming languages are
  - the most used in science and technology
  - not reliable by design (dop solutions etc)
- We plan: a CTP\(^2\)-based programming language, which
  - is based on Isabelle’s (growing!) math knowledge
  - is reliable because based on logic
  - uses Isar/Scala/jEdit as a frontend
  - extends jEdit (JavaSwing!) with a formula editor
- Potential partners
  - Isabelle at TU Munich
  - Research Inst. for Symbolic Computation (RISC), Linz
  - Inst. of Computer Languages TU Vienna
  - Inst. for Software Technology TU Graz

---

\(^1\)CAS = Computer Algebra System

\(^2\)CTP = Computer Theorem Prover
1. State of development in the ISAC-project
   - User requirements from experiments at high schools
   - Demo of the experimental ISAC-system
   - Lucas-Interpreter combines computation and deduction

2. ISAC’s architecture and future development
   - Survey on the plans
   - Details about the plans
   - What is a document model?
   - Project: CTP-based programming language

3. Summary
The ISAC-project asks for cooperation to . . .

. . . develop an educational math assistants which

- is an interactive and transparent model of math
- features independent learning by interacting with it
- supports investigation of underlying knowledge
- elicits explanatory learning by trial and error
- provides continuous learning from school to university.

’Continuous’ means: an integrated suit of software for ’all’ mathematics (i.e. Isabelle as high end)
The ISAC-project asks for cooperation to

... develop an educational math assistants which
- is an interactive and transparent model of math
- features independent learning by interacting with it
- supports investigation of underlying knowledge
- elicits explanatory learning by trial and error
- provides continuous learning from school to university.

’Continuous’ means: an integrated suit of software for ’all’ mathematics (i.e. Isabelle as high end)
The ISAC-project asks for cooperation to . . .

. . . develop an educational math assistants which
  • is an interactive and transparent model of math
  • features independent learning by interacting with it
  • supports investigation of underlying knowledge
  • elicits explanatory learning by trial and error
  • provides continuous learning from school to university.

’Continuous’ means: an integrated suit of software for ’all’ mathematics (i.e. Isabelle as high end)
The ISAC-project asks for cooperation to . . .

. . . develop an educational math assistants which
  • is an interactive and transparent model of math
  • features independent learning by interacting with it
  • supports investigation of underlying knowledge
  • elicits explanatory learning by trial and error
  • provides continuous learning from school to university.

’Continuous’ means: an integrated suit of software for ’all’ mathematics (i.e. Isabelle as high end)
The ISAC-project asks for cooperation to...

...develop an educational math assistants which
- is an interactive and transparent model of math
- features independent learning by interacting with it
- supports investigation of underlying knowledge
- elicits explanatory learning by trial and error
- provides continuous learning from school to university.

'Continuous' means: an integrated suit of software for 'all' mathematics (i.e. Isabelle as high end)
The ISAC-project asks for cooperation to . . .

. . . develop an educational math assistants which

• is an interactive and transparent model of math
• features independent learning by interacting with it
• supports investigation of underlying knowledge
• elicits explanatory learning by trial and error
• provides continuous learning from school to university.

’Continuous’ means: an integrated suit of software for ’all’ mathematics (i.e. Isabelle as high end)
The ISAC-project asks for cooperation to . . .

. . . develop an educational math assistants which

- is an interactive and transparent model of math
- features independent learning by interacting with it
- supports investigation of underlying knowledge
- elicits explanatory learning by trial and error
- provides continuous learning from school to university.

’Continuous’ means: an integrated suit of software for ’all’ mathematics (i.e. Isabelle as high end)
Thank you for attention!

F. Haftmann, C. Kaliszyk, W. Neuper

*CTP-based programming languages*.

ACM Communications in Computer Algebra 44(1/2), 2010.

Workshop THedu’11 at CADE

*(THeorem prover components for EDUcational SW)*


Initiative for an Open Source Formula Editor:

http://www.ist.tugraz.at/projects/isac/www/content/status.html#formedit