Geometry Construction Languages

guide User-Interaction

by Lucas-Interpretation

A case study, work in progress

Walther Neuper

Institute for Softwaretechnology
Graz University of Technology

3rd Workshop on Formal and Automated Theorem Proving and Applications
February 4-5, 2011
Belgrade, Serbia
1. **Transfer experiences from calculations**
   - Lucas-Interpretation in calculations
   - Requirements for tutoring software

2. **Geometry construction language (GCL) — Lucas-Interpreter**
   - Example program in Belgrade GCL
   - Specification separated from program
   - Program statements and area method
   - Lucas-interpretation for GCL
   - Checking user-input

3. **Summary**
1. Transfer experiences from calculations . . .
   Lucas-Interpretation in calculations
   Requirements for tutoring software

2. Geometry construction language (GCL) —
   Lucas-Interpreter
   Example program in Belgrade GCL
   Specification separated from program
   Program statements and area method
   Lucas-interpretation for GCL
   Checking user-input

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

Proving and Programming
Programs usually *produce* output

Isabelle/Isar
logical operating system
(contexts etc)

**i n t e r p r e t e r**

specification
program
output

Production
Lucas-Interpreter is a debugger in single-stepping mode.
Given a specification and a program ...

Authoring

**worksheet**

**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:  
\( y(0) = 0 \)  
\( \Delta y'(0) = 0 \)  
\( \Delta V(0) = q \cdot L \)  
\( \Delta M_b(L) = 0 \)

**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**
Lucas Interpreter
Walther Neuper

Calculations
Lucas Interpreter
Requirements

GCL–LucIn
Example
Specification
Area method
Lucas-Interpreter
Check input

Summary

... tutoring starts with precondition fulfilled

worksheet
Problem (B, bendl)

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
Breakpoint 1: user accepts or updates or inputs

Tutoring
Breakpoint 2: user accepts *or* updates *or* inputs

**Worksheet**

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

**Input**
- function q,
- Length L

**Pre**
- q is_integrable
- \(\Delta L > 0\)

**Output**
- 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)

**Program**

- \(\text{Script B (q, L, v, Cs)} = \)
- \(\text{LET}\)
- \(\text{funs = Subproblem (thy, pbl, met)} q, L, v\)
- \(\text{equis = Subproblem ...}\)
- \(\text{sols = Subproblem ...}\)
- \(\text{B = Take (LAST funs)}\)
- \(\text{B = ((Substitute sols)@ (Rewrite_Set poly)) B}\)
- \(\text{IN B}\)
Breakpoint 3: user accepts or updates or inputs

Tutoring

Worksheet

Problem (B, bendl)
Problem (B, load2bl)
Q(x) = c·q·x, M(x) = ...
Pre: q is integrable
\( \Delta L > 0 \)
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)

Interpreter (P.L)

Summary

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
Breakpoint 4: user accepts *or* updates *or* inputs
Breakpoint 5: user accepts or updates or inputs

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

dialog module

Isabelle/Isar
logical operating system
(contexts etc)

tutorial

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

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
Problem solved
with postcondition fulfilled

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 ...
y(x) = (q·L^2)/(4·EI) . X^2
- (q·L)/(6·EI) . x^3
+ q/(24·EI) . 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
Outline

1 Transfer experiences from calculations . . .
   Lucas-Interpretation in calculations
   Requirements for tutoring software

2 Geometry construction language (GCL) —
   Lucas-Interpreter
   Example program in Belgrade GCL
   Specification separated from program
   Program statements and area method
   Lucas-interpretation for GCL
   Checking user-input

3 Summary
A Lucas-Interpreter . . .

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

2. Check user input as generous and liberal as possible. . . provides provers with **logical context** of statements. Checking user-input is: prove derivability from context.

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

...and fulfills these **requirements** for tutoring
A Lucas-Interpreter . . .

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

2. **Check user input** as generous and liberal as possible. 
   
   . . . provides provers with **logical context** of statements. Checking user-input is: prove derivability from context.

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

...and fulfills these **requirements** for tutoring
A Lucas-Interpreter . . .

1. **Guide the user** step by step towards a solution. . . steps from **breakpoint** to **breakpoint** in a program. The user accepts or updates or inputs (dialog module!)
2. **Check user input** as generous and liberal as possible. . . provides provers with **logical context** of statements. Checking user-input is: prove derivability from context.
3. **Explain steps** on request by the user. . . interpretes **human-readable** knowledge of Isabelle. Knowledge shall be interlinked with a mathematics wiki.

...and fulfills these **requirements** for tutoring.
A Lucas-Interpreter . . .

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

2. Check user input as generous and liberal as possible.
   . . . provides provers with **logical context** of statements.
   Checking user-input is: prove derivability from context.

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

...and fulfills these **requirements** for tutoring
A Lucas-Interpreter . . .

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

2. Check user input as generous and liberal as possible.
   . . . provides provers with **logical context** of statements.
   Checking user-input is: prove derivability from context.

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

...and fulfills these **requirements** for tutoring
A Lucas-Interpreter . . .

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

2. **Check user input** as generous and liberal as possible. . . provides provers with **logical context** of statements. Checking user-input is: prove derivability from context.

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

...and fulfills these **requirements** for tutoring
A Lucas-Interpreter . . .

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

2. **Check user input** as generous and liberal as possible. 
   . . . provides provers with **logical context** of statements. 
   Checking user-input is: prove derivability from context.

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

...and fulfills these **requirements** for tutoring
1. Transfer experiences from calculations . . .
   Lucas-Interpretation in calculations
   Requirements for tutoring software

2. Geometry construction language (GCL) —
   Lucas-Interpreter
   Example program in Belgrade GCL
   Specification separated from program
   Program statements and area method
   Lucas-interpretation for GCL
   Checking user-input

3. Summary
Example program in GCL

program circum_center

    free points
    01  point A 10 10
    02  point B 40 10
    03  point C 30 40

    perpendicular bisectors of the sides
    04  med a B C
    05  med b A C

    intersection of the bisectors
    06  intersec O a b

    drawing the circumcircle of the triangle ABC
    07  drawcircle O A
1 Transfer experiences from calculations ... Lucas-Interpretation in calculations
   Requirements for tutoring software

2 Geometry construction language (GCL) —
   Lucas-Interpreter
   Example program in Belgrade GCL
   Specification separated from program
   Program statements and area method
   Lucas-interpretation for GCL
   Checking user-input

3 Summary
### Specification and program

01  point A 10 10
02  point B 40 10
03  point C 30 40

00 specification circum_center
00  input : A, B, C
00  precond : A ≠ B ∧ B ≠ C ∧ C ≠ A ∧ ¬collinear A B C
00  output : O
00  postcond: \( OA = OB \land OB = OC \land OC = OA \)

00 program circum_center A B C
04  med a B C
05  med b A C
06  intersec O a b
07  drawcircle O A
Translation to area method

01 point A 10 10
02 point B 40 10
03 point C 30 40

00 specification circum_center
00 input : A, B, C
00 precond : \( P_{ABA} \neq 0, P_{BCB} \neq 0, P_{CAC} \neq 0, \)
00 \( S_{ABC} \neq 0 \)
00 output : O
00 postcond: \( OA = OB, OB = OC, OC = OA \)

00 program circum_center A B C
04 med a B C
05 med b A C
06 intersec O a b
07 drawcircle O A
Outline

1. Transfer experiences from calculations . . .
   Lucas-Interpretation in calculations
   Requirements for tutoring software

2. Geometry construction language (GCL) —
   Lucas-Interpreter
   Example program in Belgrade GCL
   Specification separated from program
   Program statements and area method
   Lucas-interpretation for GCL
   Checking user-input

3. Summary
Program and area method

00 program circum_center A B C
ctxt_0 = { A \neq B, B \neq C, C \neq A, \neg \text{collinear } A B C }

04 med a B C
ctxt_1 = ctxt_0 \cup \{ BM_1 \parallel BC, \frac{BM_1}{BC} = \frac{1}{2}, M_1 N_1 \perp BC, \frac{4S_{M_1BN_1}}{P_{M_1BM_1}} = 1 \} 

05 med b A C
ctxt_2 = ctxt_1 \cup \{ AM_2 \parallel AC, \frac{AM_2}{AC} = \frac{1}{2}, M_2 N_2 \perp AC, \frac{4S_{M_2AN_2}}{P_{M_2AM_2}} = 1 \} 

06 intersec O a b
ctxt_3 = ctxt_2 \cup \{ \text{collinear } N_1 M_1 O, \text{collinear } N_2 M_2 O \} 

07 drawcircle O A
ctxt_4 = ctxt_3 \cup \{ OA = OB, OB = OC, OC = OA \}
Program and area method

00 program circum_center A B C
ctxt₀ = \{ A \neq B, B \neq C, C \neq A, \neg\text{collinear } A B C \} 

04 med a B C
ctxt₁ = ctxt₀ \cup \{ BM₁ \parallel BC, \frac{BM₁}{BC} = \frac{1}{2}, M₁N₁ \perp BC, \frac{4S_{M₁B1N₁}}{PM₁BM₁} = 1 \} 

05 med b A C
ctxt₂ = ctxt₁ \cup \{ AM₂ \parallel AC, \frac{AM₂}{AC} = \frac{1}{2}, M₂N₂ \perp AC, \frac{4S_{M₂A2N₂}}{PM₂AM₂} = 1 \} 

06 intersec O a b
ctxt₃ = ctxt₂ \cup \{ \text{collinear } N₁ M₁ O, \text{collinear } N₂ M₂ O \} 

07 drawcircle O A
ctxt₄ = ctxt₃ \cup \{ OA = OB, OB = OC, OC = OA \}
Program and area method

00  program  circum_center  A  B  C
ctxt₀ = \{  A ≠ B, B ≠ C, C ≠ A, \neg\text{collinear}  A  B  C  \}\}

04  med  a  B  C
ctxt₁ = ctxt₀ ∪ \{  BM₁ \parallel BC,  \frac{BM₁}{BC} = \frac{1}{2},  M₁N₁ \perp BC,  \frac{4S_{M₁BN₁}}{P_{M₁BM₁}} = 1  \}\}

05  med  b  A  C
ctxt₂ = ctxt₁ ∪ \{  AM₂ \parallel AC,  \frac{AM₂}{AC} = \frac{1}{2},  M₂N₂ \perp AC,  \frac{4S_{M₂AN₂}}{P_{M₂AM₂}} = 1  \}\}

06  intersec  O  a  b
ctxt₃ = ctxt₂ ∪ \{  \text{collinear}  N₁  M₁  O,  \text{collinear}  N₂  M₂  O  \}\}

07  drawcircle  O  A
ctxt₄ = ctxt₃ ∪ \{  OA = OB,  OB = OC,  OC = OA  \}\}
Program and area method

00 program circum_center A B C
ctxt₀ = \{ A \neq B, B \neq C, C \neq A, \neg \text{collinear } A B C \} 

04 \text{med } a \ B \ C
ctxt₁ = ctxt₀ \cup \{ BM₁ \parallel BC, \frac{BM₁}{BC} = \frac{1}{2}, M₁N₁ \perp BC, \frac{4S_{M₁BN₁}}{P_{M₁BM₁}} = 1 \} 

05 \text{med } b \ A \ C
ctxt₂ = ctxt₁ \cup \{ AM₂ \parallel AC, \frac{AM₂}{AC} = \frac{1}{2}, M₂N₂ \perp AC, \frac{4S_{M₂AN₂}}{P_{M₂AM₂}} = 1 \} 

06 \text{intersec } O \ a \ b
ctxt₃ = ctxt₂ \cup \{ \text{collinear } N₁ \ M₁ \ O, \text{collinear } N₂ \ M₂ \ O \} 

07 \text{drawcircle } O \ A
ctxt₄ = ctxt₃ \cup \{ OA = OB, OB = OC, OC = OA \}
Program and area method

00  program circum_center A B C
ctxt_0 = \{ A \not= B, B \not= C, C \not= A, \neg \text{collinear } A B C \} \\

04  \text{med } a \ B \ C \\
ctxt_1 = ctxt_0 \cup \{ BM_1 \parallel BC, \frac{BM_1}{BC} = \frac{1}{2}, M_1 N_1 \perp BC, \frac{4S_{M_1BN_1}}{P_{M_1BM_1}} = 1 \} \\

05  \text{med } b \ A \ C \\
ctxt_2 = ctxt_1 \cup \{ AM_2 \parallel AC, \frac{AM_2}{AC} = \frac{1}{2}, M_2 N_2 \perp AC, \frac{4S_{M_2AN_2}}{P_{M_2AM_2}} = 1 \} \\

06  \text{intersec } O \ a \ b \\
ctxt_3 = ctxt_2 \cup \{ \text{collinear } N_1 \ M_1 \ O, \text{collinear } N_2 \ M_2 \ O \} \\

07  \text{drawcircle } O \ A \\
ctxt_4 = ctxt_3 \cup \{ \overline{OA} = \overline{OB}, \overline{OB} = \overline{OC}, \overline{OC} = \overline{OA} \}
Program and area method

00 program circum_center A B C
cxtx_0 = { A ≠ B, B ≠ C, C ≠ A, ¬collinear A B C }

04 med a B C
cxtx_1 = cxtx_0 ∪ { BM_1 || BC, \frac{BM_1}{BC} = \frac{1}{2}, M_1 N_1 ⊥ BC, \frac{4S_{M_1 B N_1}}{P_{M_1 B M_1}} = 1 }

05 med b A C
cxtx_2 = cxtx_1 ∪ { AM_2 || AC, \frac{AM_2}{AC} = \frac{1}{2}, M_2 N_2 ⊥ AC, \frac{4S_{M_2 A N_2}}{P_{M_2 A M_2}} = 1 }

06 intersec O a b
cxtx_3 = cxtx_2 ∪ { collinear N_1 M_1 O, collinear N_2 M_2 O }

07 drawcircle O A
cxtx_4 = cxtx_3 ∪ { OA = OB, OB = OC, OC = OA }
Outline

1. Transfer experiences from calculations . . .
   Lucas-Interpretation in calculations
   Requirements for tutoring software

2. Geometry construction language (GCL) —
   Lucas-Interpreter
   Example program in Belgrade GCL
   Specification separated from program
   Program statements and area method
   Lucas-interpretation for GCL
   Checking user-input

3. Summary
Given a program and a specification . . .

\texttt{program circ A B C}
\texttt{med a B C}
\texttt{med b A C}
\texttt{intersec O a b}
\texttt{drawcircle O A}
... interpretation initialise the context with precondition

Isabelle/Isar

Lucas-Interpreter

context =

\[ A \neq B, B \neq C, C \neq A, \neg \text{collinear } A B C \]

program

\[ \text{circ } A B C \]

\[ \text{med } a B C \]

\[ \text{med } b A C \]

\[ \text{intersec } O a b \]

\[ \text{drawcircle } O A \]
Breakpoint 1: context extended with area method formulas

Breakpoint 1: context extended with area method formulas

\text{Isabelle/Isar}

\text{Lucas-Interpreter}

\text{context} = \begin{align*}
A \neq B, & \ B \neq C, \ C \neq A, \ -\text{collinear } A \ B \ C \\
BM_1 \parallel BC, & \ \frac{BM_1}{BC} = 1/2, \ M_1N_1 \perp BC, \ \frac{4S_{MBN_1}}{P_{M,BM_1}} = 1
\end{align*}

\text{program}

\text{circ } A \ B \ C
\med a \ B \ C
\med b \ A \ C
\intersec O \ a \ b
\drawcircle O \ A
Breakpoint 2: context extended with area method formulas

\[ A \neq B, B \neq C, C \neq A, -\text{collinear } A B C \]
\[ BM_1 \parallel BC, \frac{BM_1}{BC} = 1/2, M_2N_2 \perp BC, \frac{4S_{M_2BN_2}}{P_{M_2BM_1}} = 1 \]
\[ AM_2 \parallel AC, \frac{AM_2}{AC} = 1/2, M_2N_2 \perp AC, \frac{4S_{M_2AN_2}}{P_{M_2AM_2}} = 1 \]
Breakpoint 3: context extended with area method formulas

\[ A \neq B, B \neq C, C \neq A, \neg \text{collinear } A B C \]

\[ BM_1 \parallel BC, \quad \frac{BM_1}{BC} = \frac{1}{2}, \quad M_1 N_1 \perp BC, \quad \frac{4S_{M_1BN_1}}{P_{M_1BM_1}} = 1 \]

\[ AM_2 \parallel AC, \quad \frac{AM_2}{AC} = \frac{1}{2}, \quad M_2 N_2 \perp AC, \quad \frac{4S_{M_2AN_2}}{P_{M_2AM_2}} = 1 \]

\[ \text{collinear } N_1 M_1 O, \text{ collinear } N_2 M_2 O \]
Breakpoint 4: same context

\[ A \neq B, B \neq C, C \neq A, \neg \text{collinear } A B C \]

\[ BM_1 \parallel BC, \frac{BM_1}{BC} = \frac{1}{2}, M_1 N_1 \perp BC, \frac{4S_{M_1BN_1}}{P_{M_1BM_1}} = 1 \]

\[ AM_2 \parallel AC, \frac{AM_2}{AC} = \frac{1}{2}, M_2 N_2 \perp AC, \frac{4S_{M_2AN_2}}{P_{M_2AM_2}} = 1 \]

\[ \text{collinear } N_1 M_1 O, \text{ collinear } N_2 M_2 O \]

\text{Isabelle/Isar}

\text{Lucas-Interpreter}

\text{worksheet}

\text{program}

\text{program circ A B C}
\text{med a B C}
\text{med b A C}
\text{intersec O a b}
\text{drawcircle O A}
Context provides data for proof of postcondition

\[ \text{Isabelle/Isar} \]

Lucas-Interpreter

context =

\[ A \neq B, B \neq C, C \neq A, \neg \text{collinear } A B C \]

\[ BM_1 \parallel BC, \quad \frac{BM_1}{BC} = 1/2, \quad M_1N_1 \perp BC, \quad \frac{4S_{M_1BN_1}}{P_{M_1BM_1}} = 1 \]

\[ AM_2 \parallel AC, \quad \frac{AM_2}{AC} = 1/2, \quad M_2N_2 \perp AC, \quad \frac{4S_{M_2AN_2}}{P_{M_2AM_2}} = 1 \]

\[ \text{collinear } N_1 M_1 O, \text{ collinear } N_2 M_2 O \]

\[ OA = OB, \quad OB = OC, \quad OC = OA \]

\[ \text{worksheet} \]

\[ \text{program} \]

program circ A B C

med a B C

med b A C

intersec O a b
drawcircle O A
A Lucas-Interpreter provides . . .

- **logical context** for specific program statements
- a final **context for proving** the postcondition

Summary
A Lucas-Interpreter provides . . .

- **logical context** for specific program statements
- a final **context for proving** the postcondition
1 Transfer experiences from calculations . . .
   Lucas-Interpretation in calculations
   Requirements for tutoring software

2 Geometry construction language (GCL) —
   Lucas-Interpreter
   Example program in Belgrade GCL
   Specification separated from program
   Program statements and area method
   Lucas-interpretation for GCL
   Checking user-input

3 Summary
Somewhere during stepwise construction . . .

Isabelle/Isar

Lucas-Interpreter

worksheet

context =

A ≠ B, B ≠ C, C ≠ A, ¬collinear A B C

program

program circ A B C

med a B C

med b A C

intersec O a b
drawcircle O A
Somewhere during stepwise construction . . .

\[ A \neq B, B \neq C, C \neq A, \neg \text{collinear } A B C \]

\[ BM_1 \parallel BC, \frac{BM_1}{BC} = 1/2, M_3N_3 \perp BC, \frac{4S_{M_1BN_1}}{\overrightarrow{P_{M_1BM_1}}} = 1 \]
...the user inputs the next step: logical data created!

Isabelle/Isar

Lucas-Interpreter

context =

\[ A \neq B, B \neq C, C \neq A, \text{collinear } A B C \]
\[ BM_1 \parallel BC, \frac{BM_1}{BC} = 1/2, M_2N_3 \perp BC, \frac{4S_{M_2BN_3}}{P_{M_2BM_3}} = 1 \]
\[ BM_3 \parallel BA, \frac{BM_3}{BA} = 1/2, M_3N_3 \perp BA, \frac{4S_{M_2BN_3}}{P_{M_2BM_3}} = 1 \]
Lucas-Interpretation compares with subsequent contexts

```
program circ  A B C
    med a B C
    med b A C
    intersec O a b
    drawcircle O A

programworksheet
```

### Isabelle/Isar

**context =**

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

\[ BM_1 \parallel BC, \frac{BM_1}{BC} = \frac{1}{2}, M_2N_3 \perp BC, \frac{4S_{M_1BN_3}}{P_{M_1BM_1}} = 1 \]

\[ BM_3 \parallel BA, \frac{BM_3}{BA} = \frac{1}{2}, M_3N_3 \perp BA, \frac{4S_{M_3BN_3}}{P_{M_2BM_3}} = 1 \]

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

\[ A \neq B, B \neq C, C \neq A, \text{–collinear } A B C \]
\[ BM_1 \| BC, \frac{BM_1}{BC} = \frac{1}{2}, M_1 N_1 \perp BC, \frac{4S_{M_1 BN_1}}{P_{M_1 BM_1}} = 1 \]
\[ BM_3 \| BA, \frac{BM_3}{BA} = \frac{1}{2}, M_3 N_3 \perp BA, \frac{4S_{M_3 BN_3}}{P_{M_3 BM_3}} = 1 \]
\[ AM_2 \| AC, \frac{AM_2}{AC} = \frac{1}{2}, M_2 N_2 \perp AC, \frac{4S_{M_2 AN_2}}{P_{M_2 AM_2}} = 1 \]
\[ \text{collinear } N_1 M_1 O, \text{ collinear } N_2 M_2 O \]

program

- program circ \ A B C
- med a B C
- med b A C
- intersec O a b
drawcircle O A
A Lucas-Interpreter provides . . .

- **logical context** for specific program statements
- a final **context for proving** the postcondition
- logical context also for **user input**
- **several possibilities** for handling user input:
  - equality of subsequent statements
  - equality of subsequent contexts
  - *equivalence* (?) of contexts
  - *interpolants* measure ’distance’ to postcondition (?)
A Lucas-Interpreter provides . . .

- **logical context** for specific program statements
- a final **context for proving** the postcondition
- logical context also for **user input**

- several possibilities for handling user input:
  - equality of subsequent statements
  - equality of subsequent contexts
  - equivalence (?) of contexts
  - *interpolants* measure ‘distance’ to postcondition (?)
  - . . .
A Lucas-Interpreter provides . . .

- **logical context** for specific program statements
- a final **context for proving** the postcondition
- logical context also for **user input**
- **several possibilities** for handling user input:
  - equality of subsequent statements
  - equality of subsequent contexts
  - *equivalence* (?) of contexts
  - *interpolants* measure ’distance’ to postcondition (?)
A Lucas-Interpreter provides . . .

- **logical context** for specific program statements
- a final **context for proving** the postcondition
- logical context also for **user input**
- **several possibilities** for handling user input:
  - equality of subsequent statements
  - equality of subsequent contexts
  - equivalence (?) of contexts
  - *interpolants* measure ‘distance’ to postcondition (?)
  - . . .
1. Transfer experiences from calculations . . .
   Lucas-Interpretation in calculations
   Requirements for tutoring software

2. Geometry construction language (GCL) — Lucas-Interpreter
   Example program in Belgrade GCL
   Specification separated from program
   Program statements and area method
   Lucas-interpretation for GCL
   Checking user-input

3. Summary
Can Lucas-Interpret. on GCL...

1. …guide the user step by step towards a solution?
   Yes! (program coded by hand or synthesised)
   Interesting: where handle cartesian coordinates?

2. …check user input as generous as possible?
   Equivalent contexts by area method’s simplification?
   Automation in other axiom systems?

3. …explain steps on request by the user?
   Med a B C .... BM_1 || BC, \( \frac{BM_1}{BC} = \frac{1}{2} \), M_1 N_1 \perp BC, \frac{4S_{M_1 BN_1}}{P_{M_1 BM_1}} = 1
   Human readability of other axiom systems?

... Belgrade GCL — a promising R&D tool.
Can Lucas-Interpret. on GCL... 

1. **guide the user** step by step towards a solution? Yes! (program coded by hand or synthesised)
   Interesting: where handle cartesian coordinates?

2. **check user input** as generous as possible?
   Equivalent contexts by area method’s simplification?
   Automation in other axiom systems?

3. **explain steps** on request by the user?
   \[ \text{med a B C} \ldots \quad BM_1 \parallel BC, \quad \frac{BM_1}{BC} = \frac{1}{2}, \quad M_1 N_1 \perp BC, \quad 4S_{M_1BN_1} \quad \frac{4S_{M_1BN_1}}{P_{M_1BM_1}} = 1 \]
   Human readability of other axiom systems?

... Belgrade GCL — a promising R&D tool.
Can Lucas-Interpret. on GCL…

1. **...guide the user** step by step towards a solution? Yes! (program coded by hand or synthesised)
   Interesting: where handle cartesian coordinates?

2. **...check user input** as generous as possible?
   Equivalent contexts by area method’s simplification?
   Automation in other axiom systems?

3. **...explain steps** on request by the user?
   \[
   \text{med a B C} \quad \ldots \quad BM_1 \parallel BC, \quad \frac{BM_1}{BC} = \frac{1}{2}, \quad M_1 N_1 \perp BC, \quad \frac{4S_{M_1BN_1}}{P_{M_1BM_1}} = 1
   \]
   Human readability of other axiom systems?

...Belgrade GCL — a promising R&D tool.
Can Lucas-Interpret. on GCL...

1. **guide the user** step by step towards a solution? Yes! (program coded by hand or synthesised)
   Interesting: where handle cartesian coordinates?

2. **check user input** as generous as possible?
   *Equivalent* contexts by area method’s simplification?
   Automation in other axiom systems?

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

   \[ \text{med} \ a \ B \ C \ldots \ BM_1 \parallel BC, \ \frac{BM_1}{BC} = \frac{1}{2}, \ M_1 N_1 \perp BC, \ \frac{4s_{M_1BN_1}}{PM_{BM_1}} = 1 \]

   Human readability of other axiom systems?

... Belgrade GCL — a promising R&D tool.
Can Lucas-Interpret. on GCL…

1. . . guide the user step by step towards a solution? Yes! (program coded by hand or synthesised)
   Interesting: where handle cartesian coordinates?

2. . . check user input as generous as possible?
   Equivalent contexts by area method’s simplification?
   Automation in other axiom systems?

3. . . explain steps on request by the user?
   \[
   \text{med a B C} \Rightarrow BM_1 \parallel BC, \quad \frac{BM_1}{BC} = \frac{1}{2}, \quad M_1N_1 \perp BC, \quad \frac{4S_{M_1BN_1}}{P_{M_1BM_1}} = 1
   \]
   Human readability of other axiom systems?

. . . Belgrade GCL — a promising R&D tool.
Can Lucas-Interpret. on GCL...

1. **guide the user** step by step towards a solution? Yes! (program coded by hand or synthesised)
   Interesting: where handle cartesian coordinates?

2. **check user input** as generous as possible?
   *Equivalent* contexts by area method’s simplification?
   Automation in other axiom systems?

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

    \[ \text{med } a \ B \ C \ldots \ BM_1 \parallel BC, \quad \frac{BM_1}{BC} = \frac{1}{2}, \quad M_1 N_1 \perp BC, \quad \frac{4S_{M_1BN_1}}{P_{M_1BM_1}} = 1 \]

   Human readability of other axiom systems?

   ...Belgrade GCL — a promising R&D tool.
Can Lucas-Interpret. on GCL...

1. *guide the user* step by step towards a solution?
   Yes! (program coded by hand or synthesised)
   Interesting: where handle cartesian coordinates?

2. *check user input* as generous as possible?
   *Equivalent* contexts by area method’s simplification?
   Automation in other axiom systems?

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

\[
\text{med } a \ B \ C \ldots \ BM_1 \parallel BC, \ \frac{BM_1}{BC} = \frac{1}{2}, \ M_1 N_1 \perp BC, \ \frac{4S_{M_1B_1N_1}}{P_{M_1B_1}} = 1
\]

Human readability of other axiom systems?

...Belgrade GCL — a promising R&D tool.
Can Lucas-Interpret. on GCL. . .

1. . . guide the user step by step towards a solution ?
   Yes ! (program coded by hand or synthesised)
   Interesting: where handle cartesian coordinates ?

2. . . check user input as generous as possible ?
   Equivalent contexts by area method’s simplification ?
   Automation in other axiom systems ?

3. . . explain steps on request by the user ?
   \[
   \text{med} \ a \ b \ c \ldots \ BM_1 \parallel BC, \ \frac{BM_1}{BC} = \frac{1}{2}, \ M_1 N_1 \perp BC, \ P_{M_1 BM_1} = 1
   \]
   Human readability of other axiom systems ?

. . . Belgrade GCL — a promising R&D tool.
Thank you for attention!

Wiki for joint work on the questions:
https://lsiit-cnrs.unistra.fr/DG-Proofs-Construction
Workshop THedu’11 at CADE: