Geometry Construction Languages guide User-Interaction by Lucas-Interpretation

A case study, work in progress

Walther Neuper

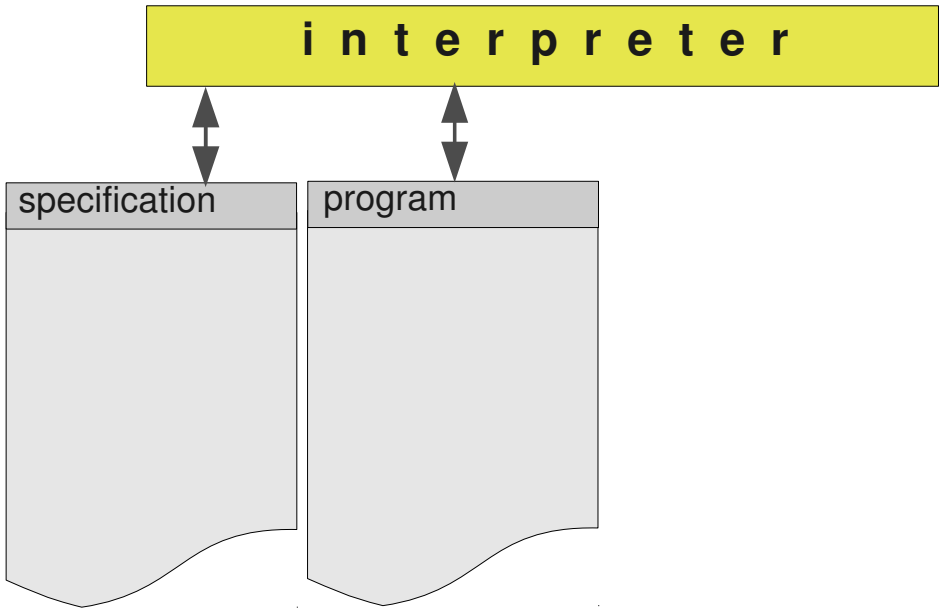
Institute for Softwaretechnology
Graz University of Technology

3^{rd} Workshop on Formal and Automated Theorem Proving and Applications
February 4-5, 2011
Belgrade, Serbia

Outline

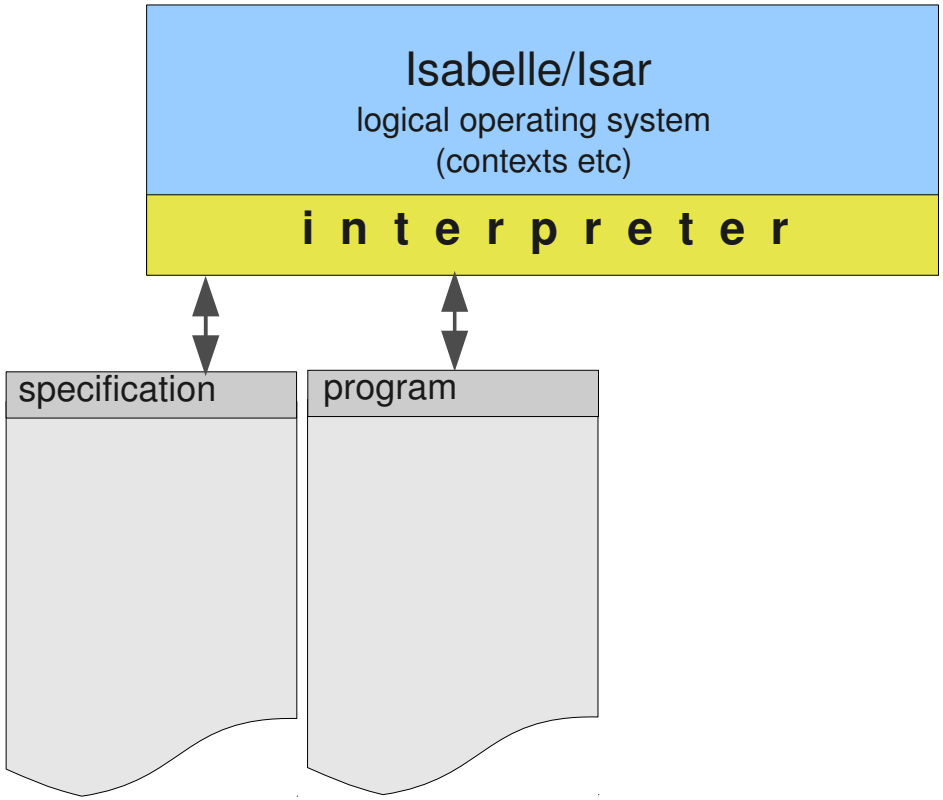
Outline

Lucas-interpretation combines *proving and programming*

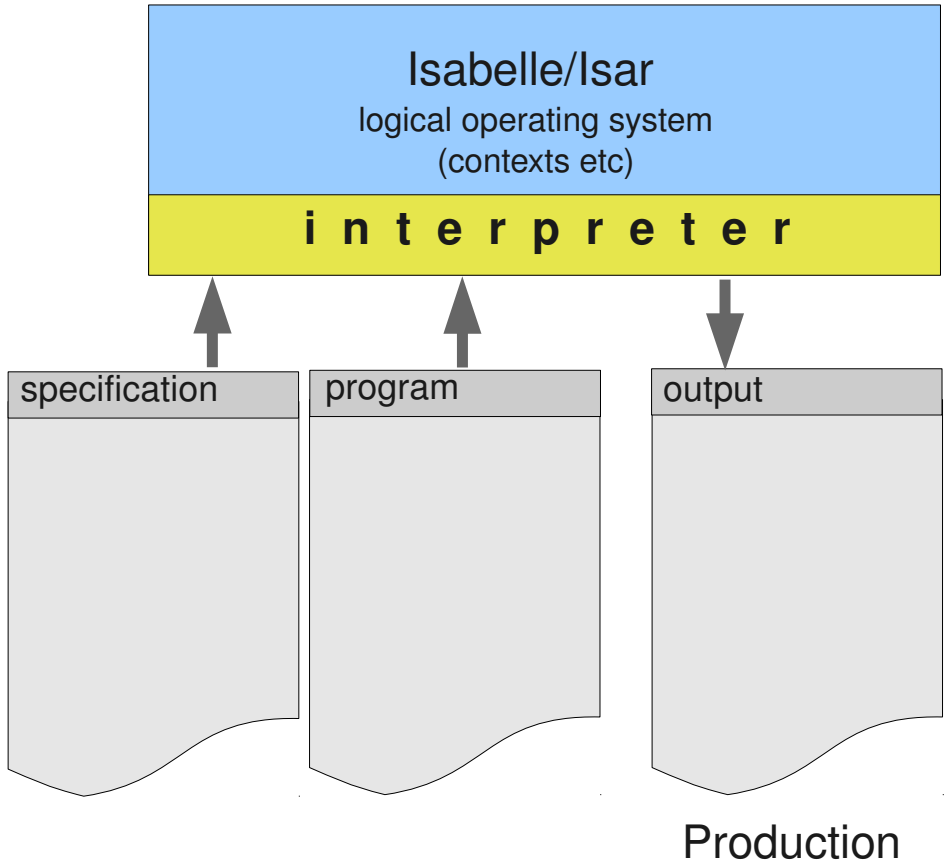


Proving and Programming

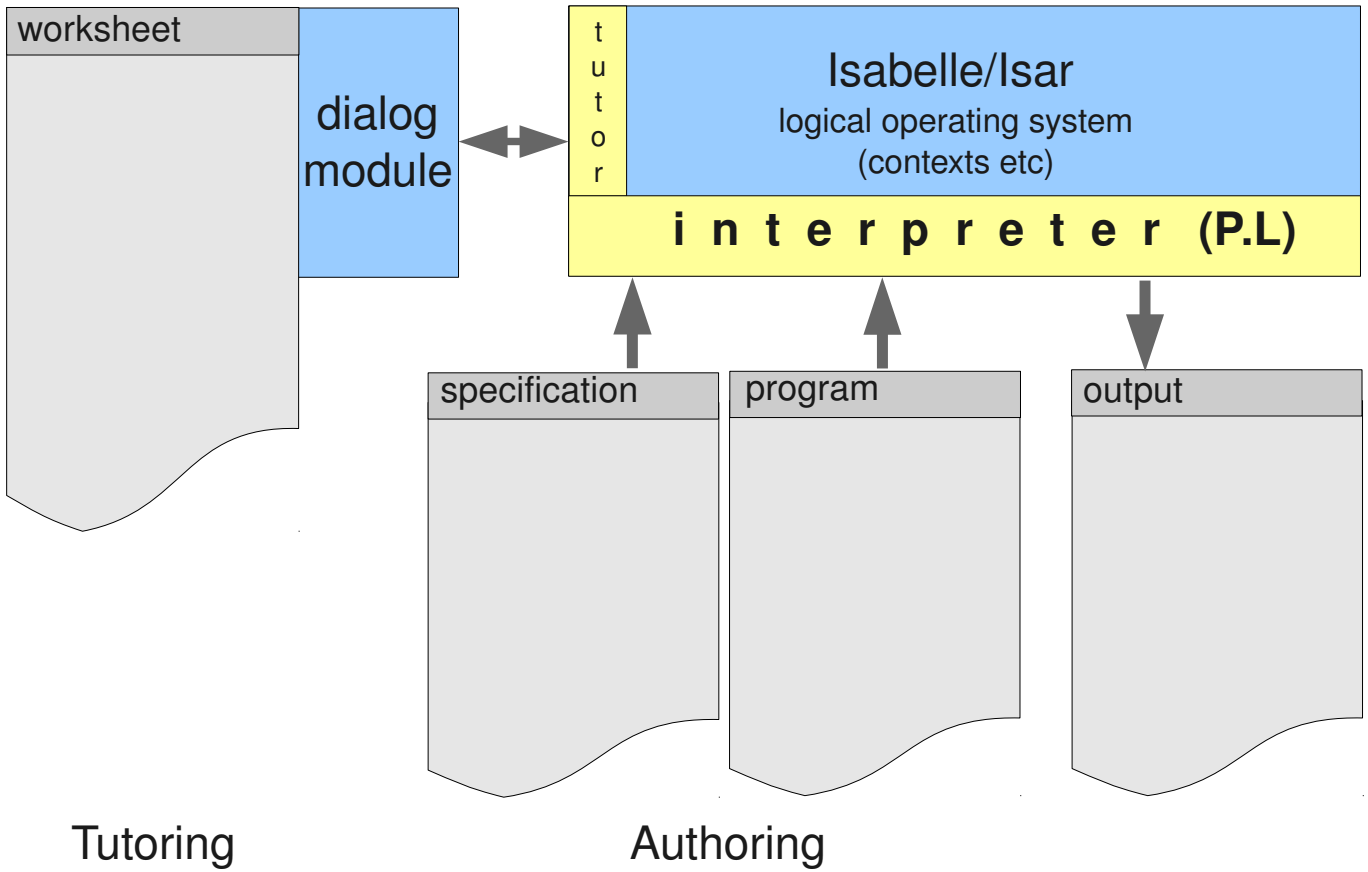
The $\mathcal{ISAC}$-prototype is based on Isabelle



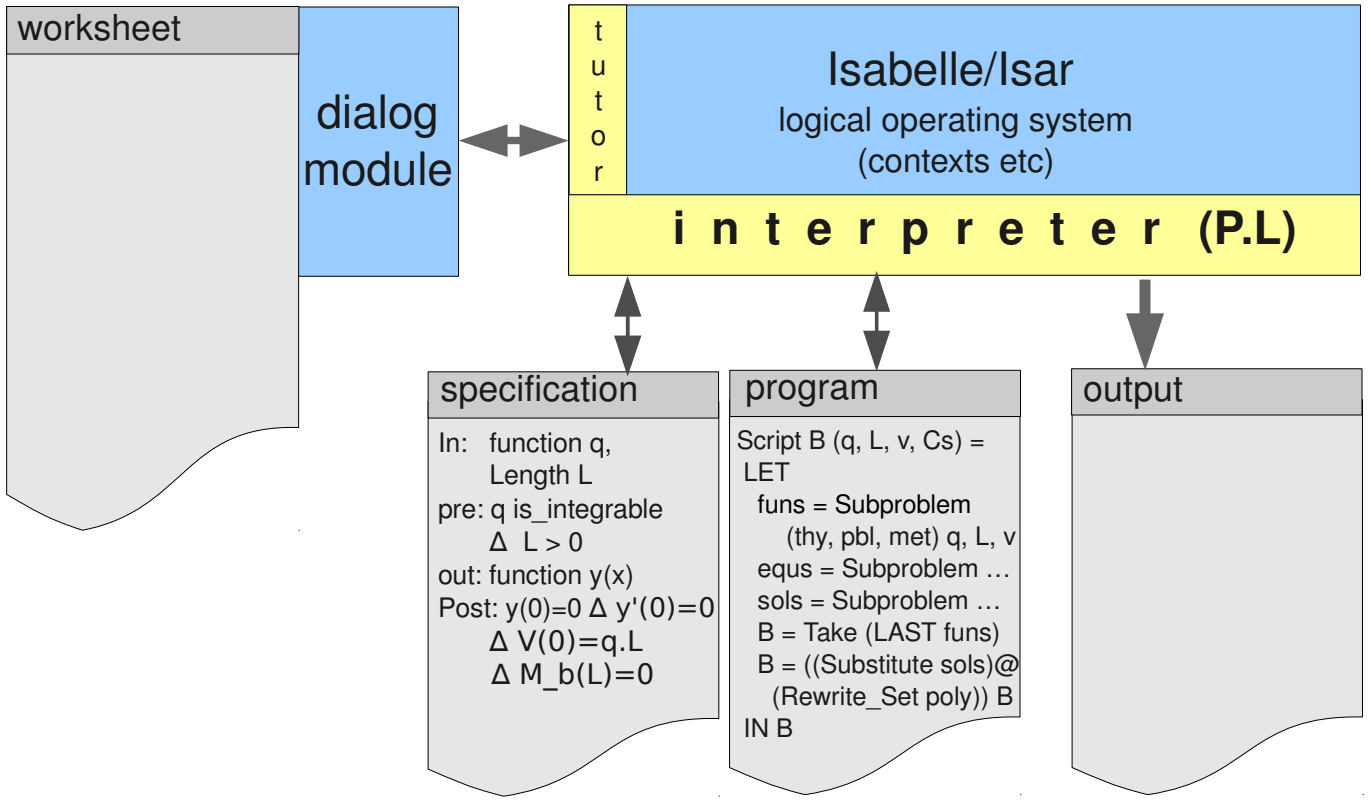
Programs usually *produce* output



Lucas-Interpreter is a debugger in single-stepping mode

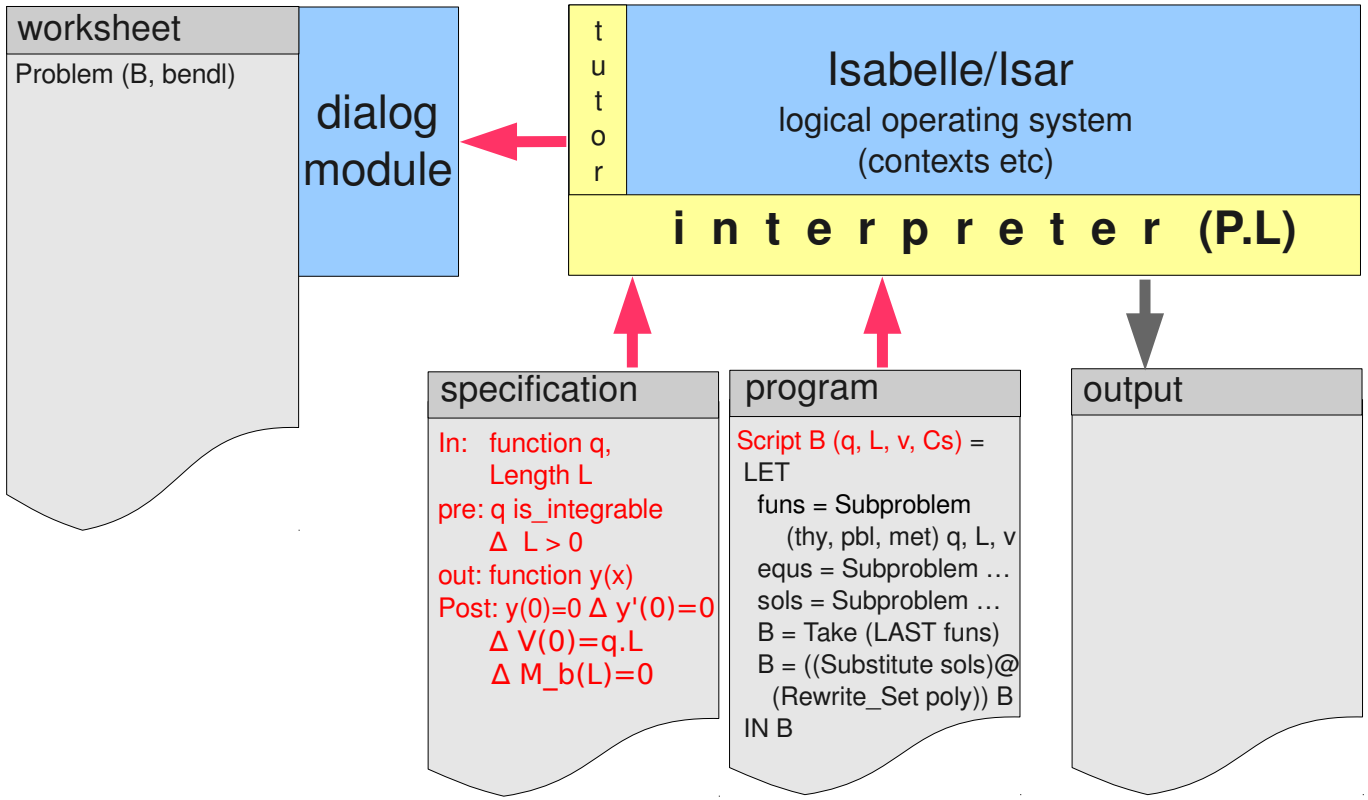


Given a specification and a program . . .



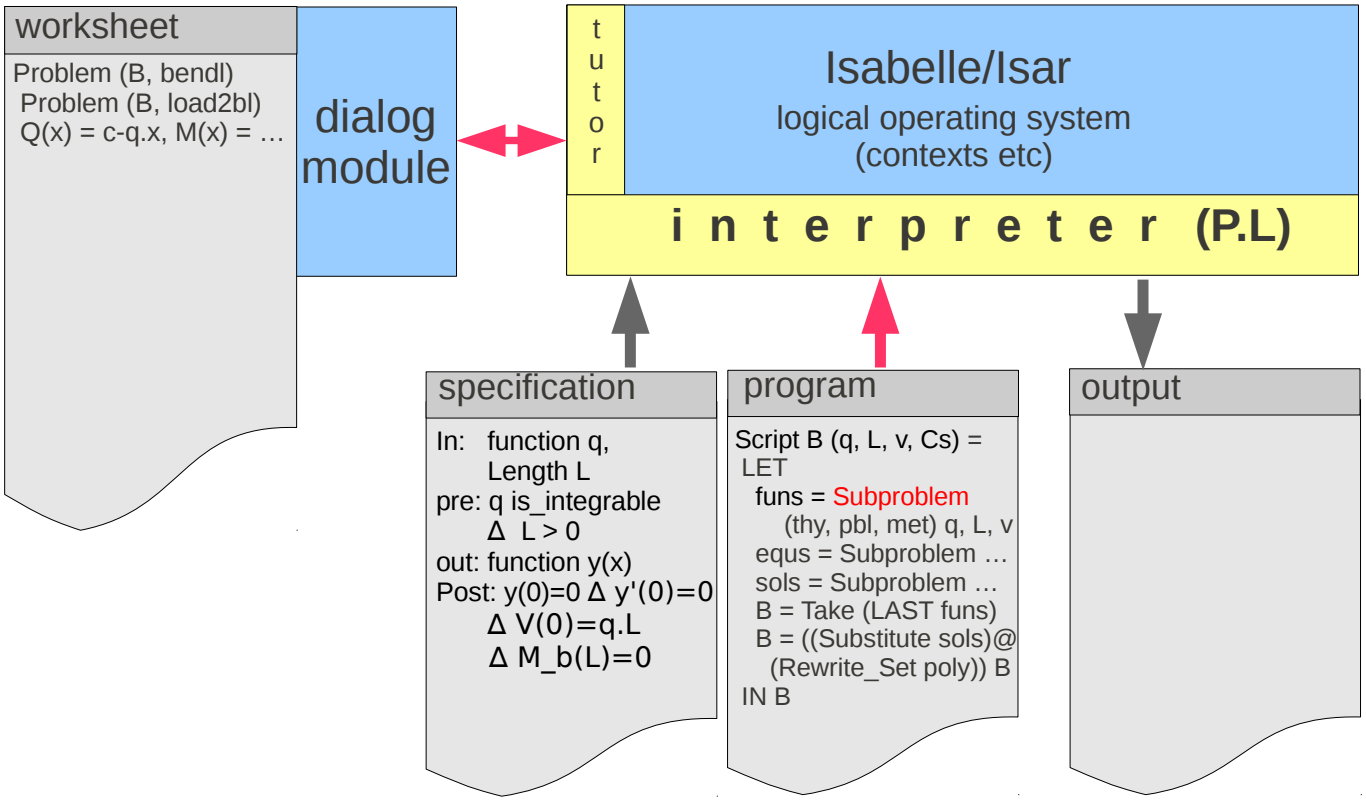
Authoring

. . . tutoring starts with precondition fulfilled



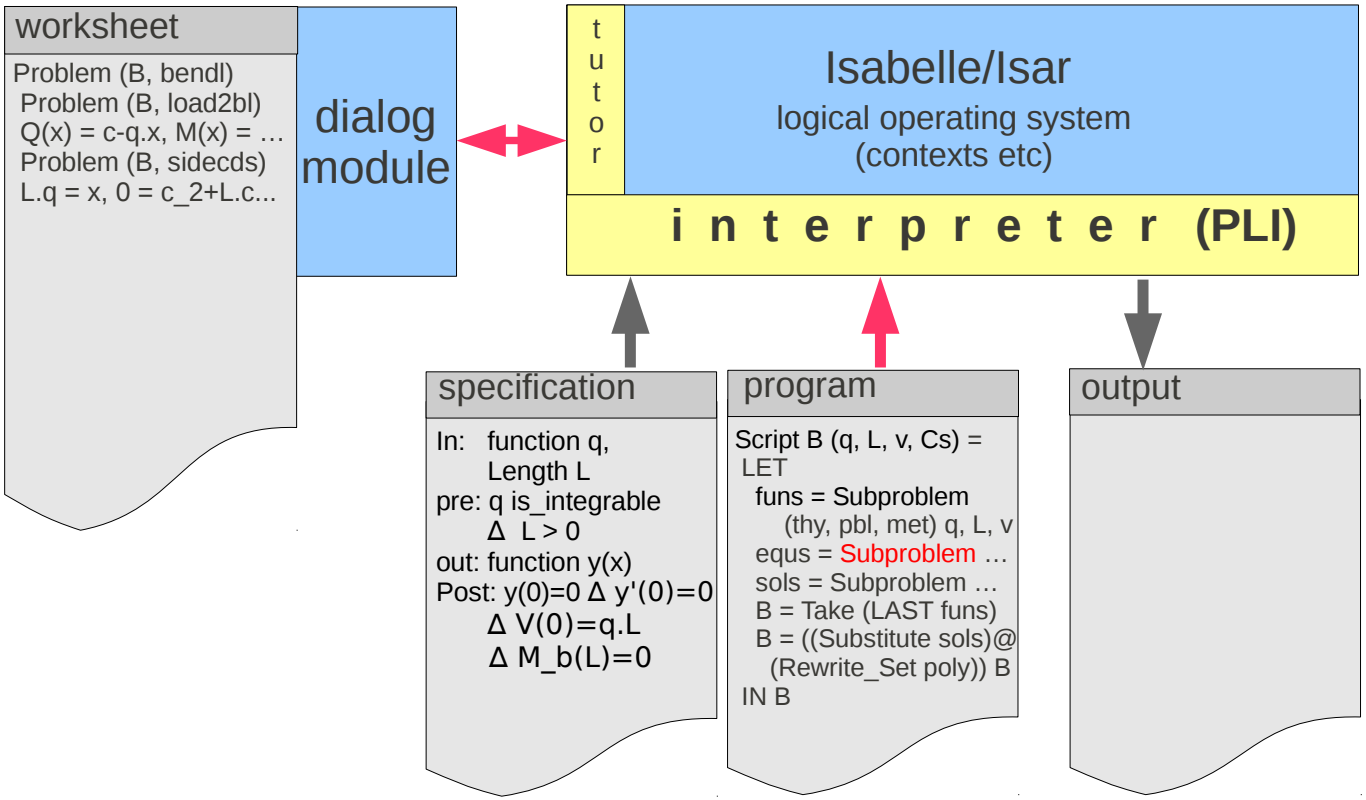
Tutoring

Breakpoint 1: user accepts *or* updates *or* inputs



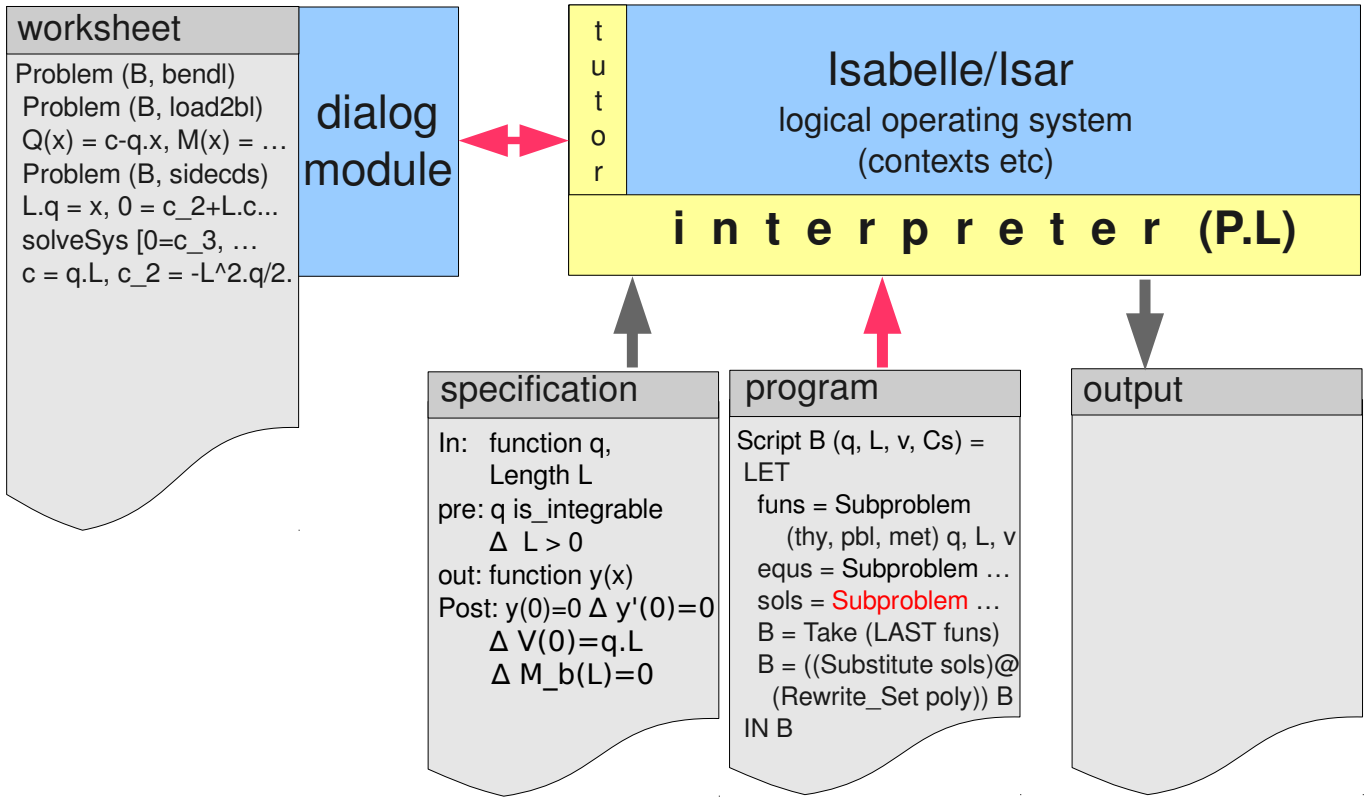
Tutoring

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



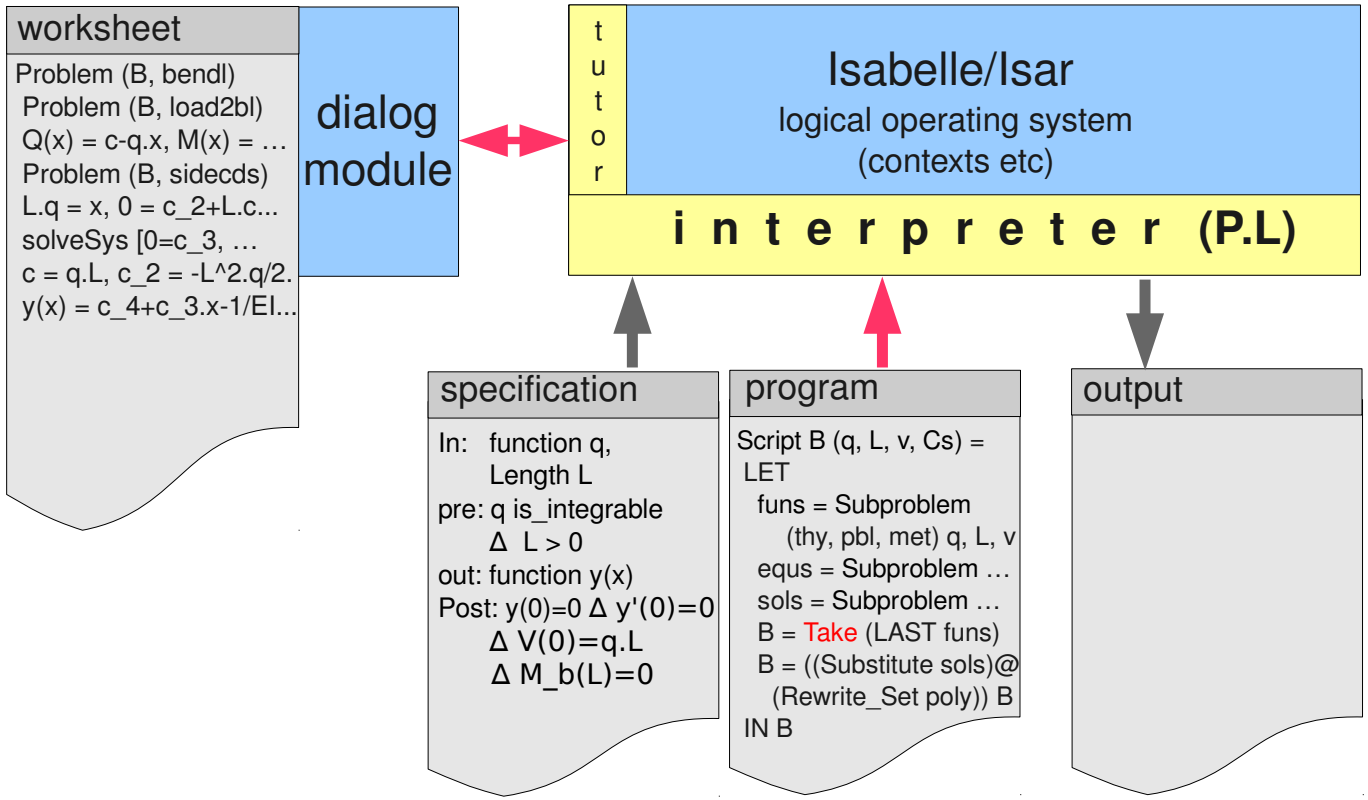
Tutoring

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



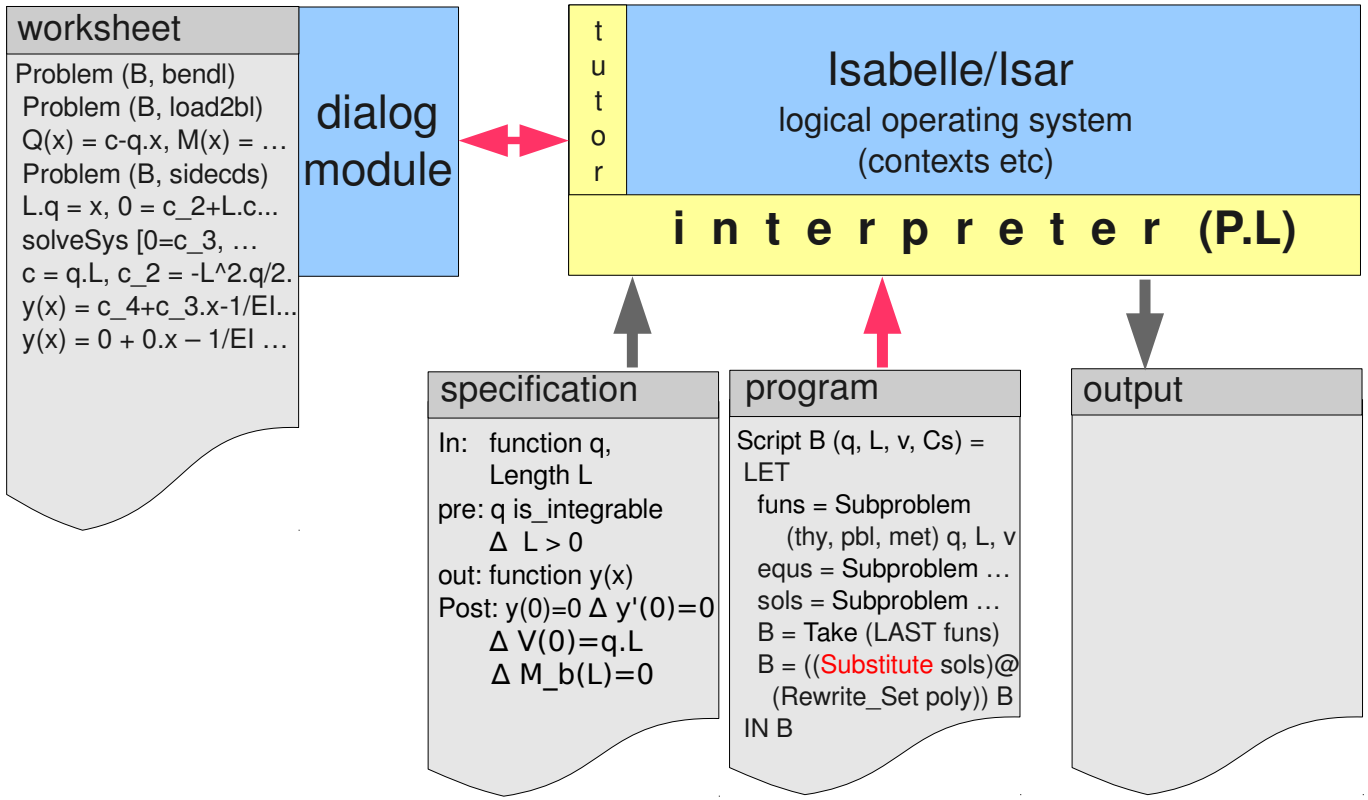
Tutoring

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



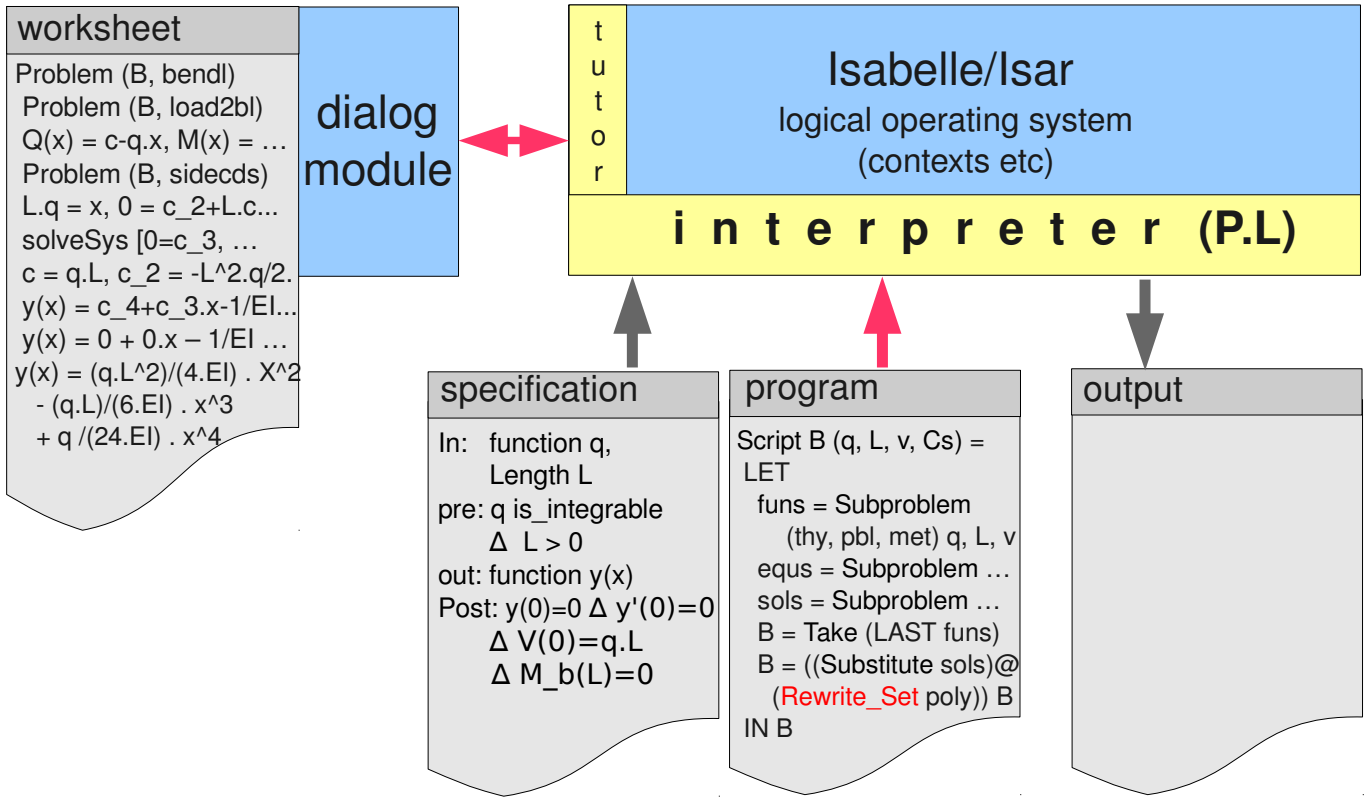
Tutoring

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



Tutoring

Problem solved with postcondition fulfilled



Tutoring

Outline

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

Outline

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
```

Outline

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 ∧
00              ¬ collinear A B C
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
```

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 ≠ 0, P_BCB ≠ 0, P_CAC ≠ 0,
00              S_ABC ≠ 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

Program and area method

00 `program circum_center A B C`

$ctxt_0 = \{ A \neq B, B \neq C, C \neq A, \neg collinear\ A\ B\ C \}$

04 `med a B C`

$ctxt_1 = ctxt_0 \cup \{ BM_1 \mid\mid BC,\ \frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2},\ M_1N_1 \perp BC,\ \frac{4\mathcal{S}_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1 \}$

05 `med b A C`

$ctxt_2 = ctxt_1 \cup \{ AM_2 \mid\mid AC,\ \frac{\overline{AM_2}}{\overline{AC}} = \frac{1}{2},\ M_2N_2 \perp AC,\ \frac{4\mathcal{S}_{M_2AN_2}}{\mathcal{P}_{M_2AM_2}} = 1 \}$

06 `intersec O a b`

$ctxt_3 = ctxt_2 \cup \{ collinear\ N_1\ M_1\ O,\ collinear\ N_2\ M_2\ O \}$

07 `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`

$\text{ctxt}_0 = \{\ A \neq B, B \neq C, C \neq A, \neg \textit{collinear}\ A\ B\ C\ \}$

04 `med a B C`

$\text{ctxt}_1 = \text{ctxt}_0 \cup \{\ BM_1 \parallel BC,\ \frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2},\ M_1N_1 \perp BC,\ \frac{4\mathcal{S}_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1\ \}$

05 `med b A C`

$\text{ctxt}_2 = \text{ctxt}_1 \cup \{\ AM_2 \parallel AC,\ \frac{\overline{AM_2}}{\overline{AC}} = \frac{1}{2},\ M_2N_2 \perp AC,\ \frac{4\mathcal{S}_{M_2AN_2}}{\mathcal{P}_{M_2AM_2}} = 1\ \}$

06 `intersec O a b`

$\text{ctxt}_3 = \text{ctxt}_2 \cup \{\ \textit{collinear}\ N_1\ M_1\ O,\ \textit{collinear}\ N_2\ M_2\ O\ \}$

07 `drawcircle O A`

$\text{ctxt}_4 = \text{ctxt}_3 \cup \{\ \overline{OA} = \overline{OB},\ \overline{OB} = \overline{OC},\ \overline{OC} = \overline{OA}\ \}$

Program and area method

00 `program circum_center A B C`

$\text{ctxt}_0 = \{\ A \neq B, B \neq C, C \neq A, \neg \textit{collinear}\ A\ B\ C\ \}$

04 `med a B C`

$\text{ctxt}_1 = \text{ctxt}_0 \cup \{\ BM_1 \parallel BC,\ \frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2},\ M_1N_1 \perp BC,\ \frac{4\mathcal{S}_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1\ \}$

05 `med b A C`

$\text{ctxt}_2 = \text{ctxt}_1 \cup \{\ AM_2 \parallel AC,\ \frac{\overline{AM_2}}{\overline{AC}} = \frac{1}{2},\ M_2N_2 \perp AC,\ \frac{4\mathcal{S}_{M_2AN_2}}{\mathcal{P}_{M_2AM_2}} = 1\ \}$

06 `intersec O a b`

$\text{ctxt}_3 = \text{ctxt}_2 \cup \{\ \textit{collinear}\ N_1\ M_1\ O,\ \textit{collinear}\ N_2\ M_2\ O\ \}$

07 `drawcircle O A`

$\text{ctxt}_4 = \text{ctxt}_3 \cup \{\ \overline{OA} = \overline{OB},\ \overline{OB} = \overline{OC},\ \overline{OC} = \overline{OA}\ \}$

Program and area method

00 `program circum_center A B C`

$\text{ctxt}_0 = \{\ A \neq B, B \neq C, C \neq A, \neg \mathit{collinear}\ A\ B\ C\ \}$

04 `med a B C`

$\text{ctxt}_1 = \text{ctxt}_0 \cup \{\ BM_1 \parallel BC,\ \frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2},\ M_1N_1 \perp BC,\ \frac{4\mathcal{S}_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1\ \}$

05 `med b A C`

$\text{ctxt}_2 = \text{ctxt}_1 \cup \{\ AM_2 \parallel AC,\ \frac{\overline{AM_2}}{\overline{AC}} = \frac{1}{2},\ M_2N_2 \perp AC,\ \frac{4\mathcal{S}_{M_2AN_2}}{\mathcal{P}_{M_2AM_2}} = 1\ \}$

06 `intersec O a b`

$\text{ctxt}_3 = \text{ctxt}_2 \cup \{\ \mathit{collinear}\ N_1\ M_1\ O,\ \mathit{collinear}\ N_2\ M_2\ O\ \}$

07 `drawcircle O A`

$\text{ctxt}_4 = \text{ctxt}_3 \cup \{\ \overline{OA} = \overline{OB},\ \overline{OB} = \overline{OC},\ \overline{OC} = \overline{OA}\ \}$

Program and area method

00 `program circum_center A B C`

$\text{ctxt}_0 = \{\ A \neq B, B \neq C, C \neq A, \neg \mathit{collinear}\ A\ B\ C\ \}$

04 `med a B C`

$\text{ctxt}_1 = \text{ctxt}_0 \cup \{\ BM_1 \parallel BC,\ \frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2},\ M_1N_1 \perp BC,\ \frac{4\mathcal{S}_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1\ \}$

05 `med b A C`

$\text{ctxt}_2 = \text{ctxt}_1 \cup \{\ AM_2 \parallel AC,\ \frac{\overline{AM_2}}{\overline{AC}} = \frac{1}{2},\ M_2N_2 \perp AC,\ \frac{4\mathcal{S}_{M_2AN_2}}{\mathcal{P}_{M_2AM_2}} = 1\ \}$

06 `intersec O a b`

$\text{ctxt}_3 = \text{ctxt}_2 \cup \{\ \mathit{collinear}\ N_1\ M_1\ O,\ \mathit{collinear}\ N_2\ M_2\ O\ \}$

07 `drawcircle O A`

$\text{ctxt}_4 = \text{ctxt}_3 \cup \{\ \overline{OA} = \overline{OB},\ \overline{OB} = \overline{OC},\ \overline{OC} = \overline{OA}\ \}$

Program and area method

00 `program circum_center A B C`

$\text{ctxt}_0 = \{\ A \neq B, B \neq C, C \neq A, \neg \mathit{collinear}\ A\ B\ C\ \}$

04 `med a B C`

$\text{ctxt}_1 = \text{ctxt}_0 \cup \{\ BM_1 \parallel BC,\ \frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2},\ M_1N_1 \perp BC,\ \frac{4\mathcal{S}_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1\ \}$

05 `med b A C`

$\text{ctxt}_2 = \text{ctxt}_1 \cup \{\ AM_2 \parallel AC,\ \frac{\overline{AM_2}}{\overline{AC}} = \frac{1}{2},\ M_2N_2 \perp AC,\ \frac{4\mathcal{S}_{M_2AN_2}}{\mathcal{P}_{M_2AM_2}} = 1\ \}$

06 `intersec O a b`

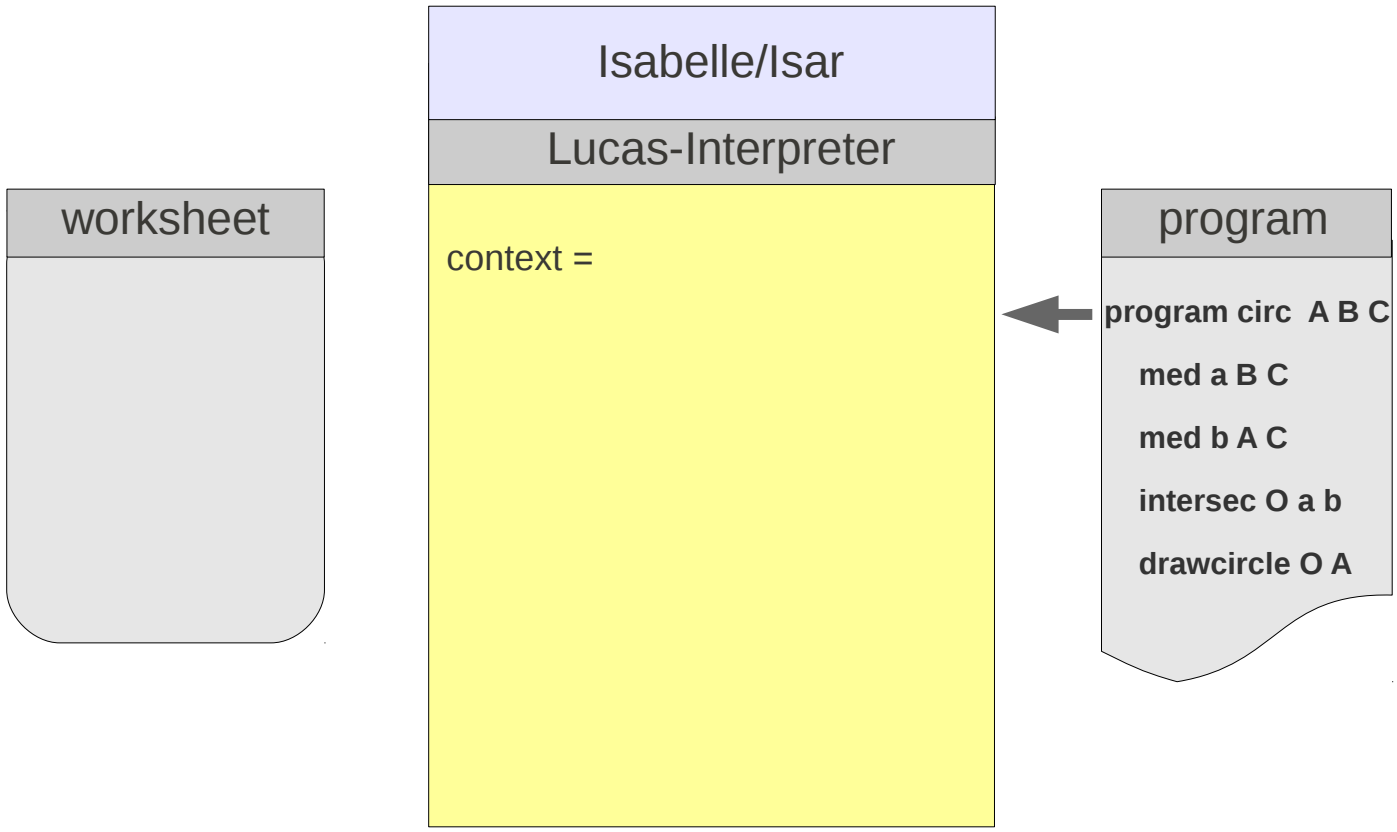
$\text{ctxt}_3 = \text{ctxt}_2 \cup \{\ \mathit{collinear}\ N_1\ M_1\ O,\ \mathit{collinear}\ N_2\ M_2\ O\ \}$

07 `drawcircle O A`

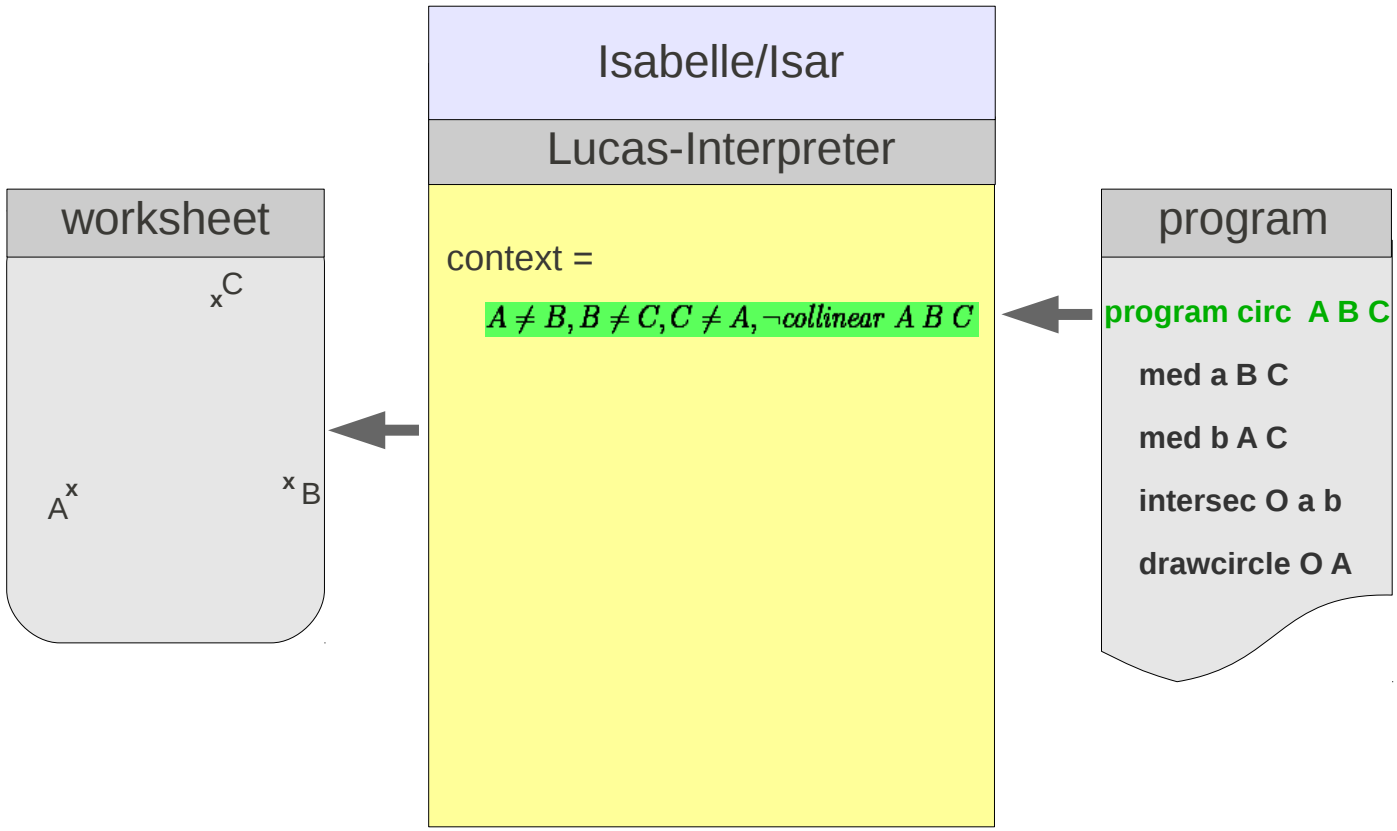
$\text{ctxt}_4 = \text{ctxt}_3 \cup \{\ \overline{OA} = \overline{OB},\ \overline{OB} = \overline{OC},\ \overline{OC} = \overline{OA}\ \}$

Outline

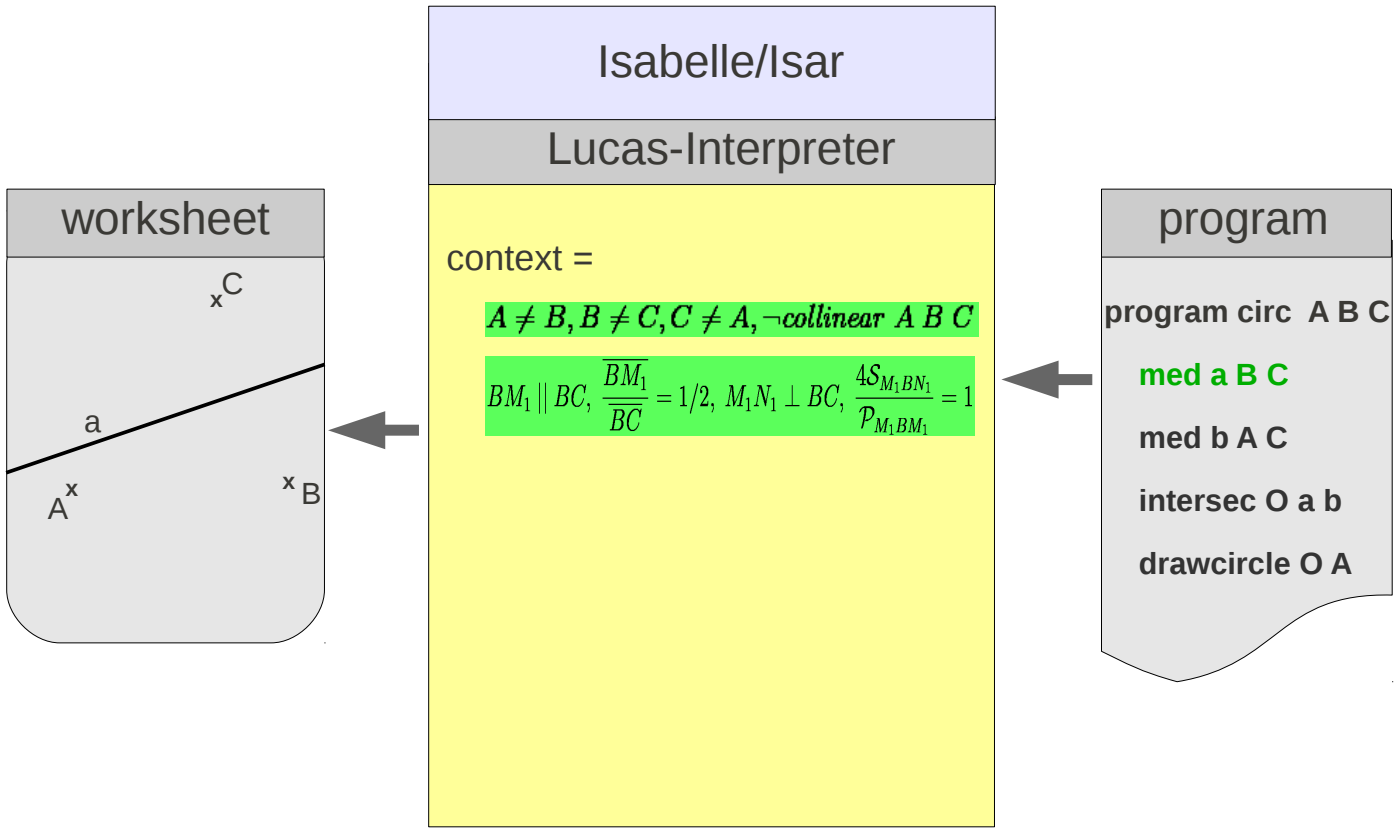
Given a program and a specification . . .



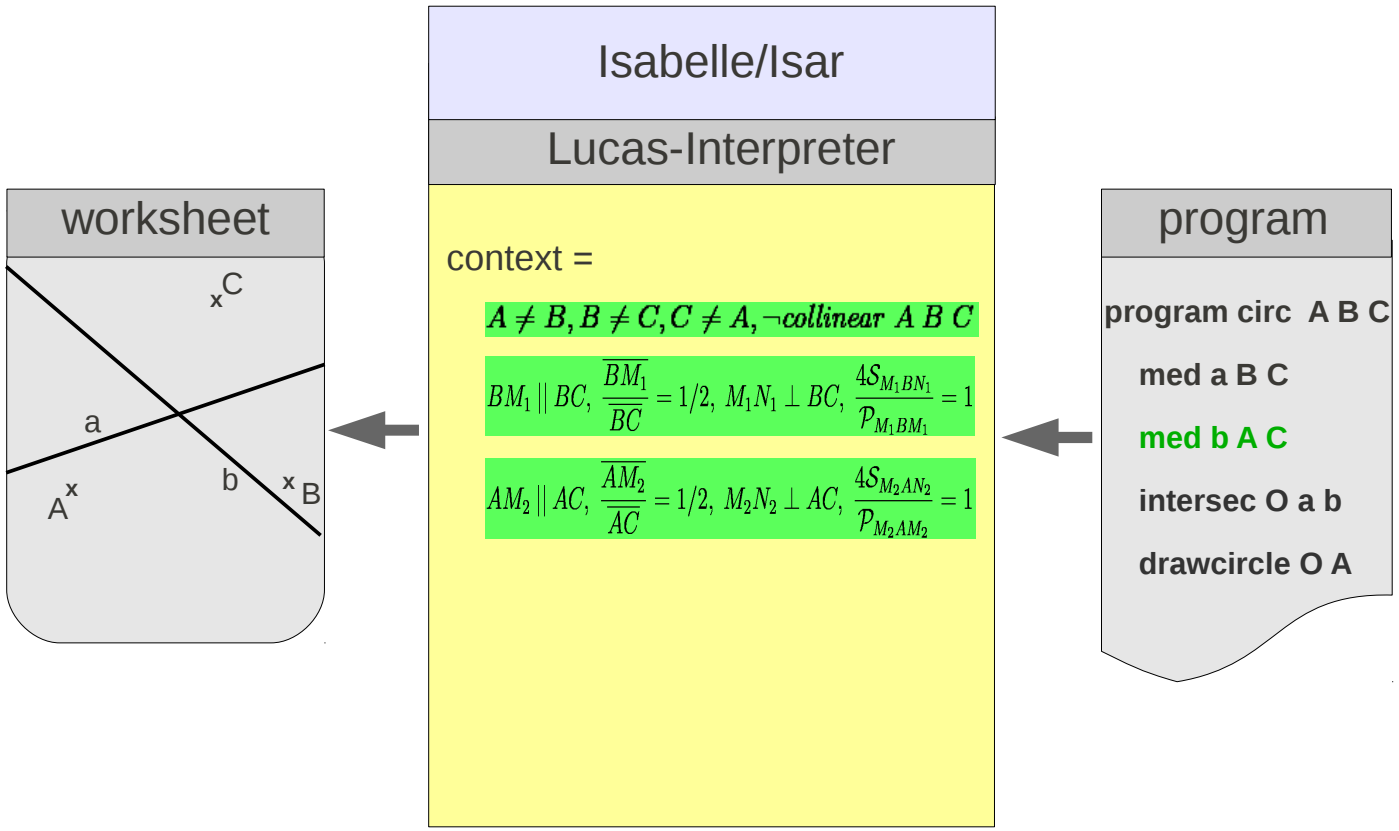
. . . interpretation initialises the context with precondition



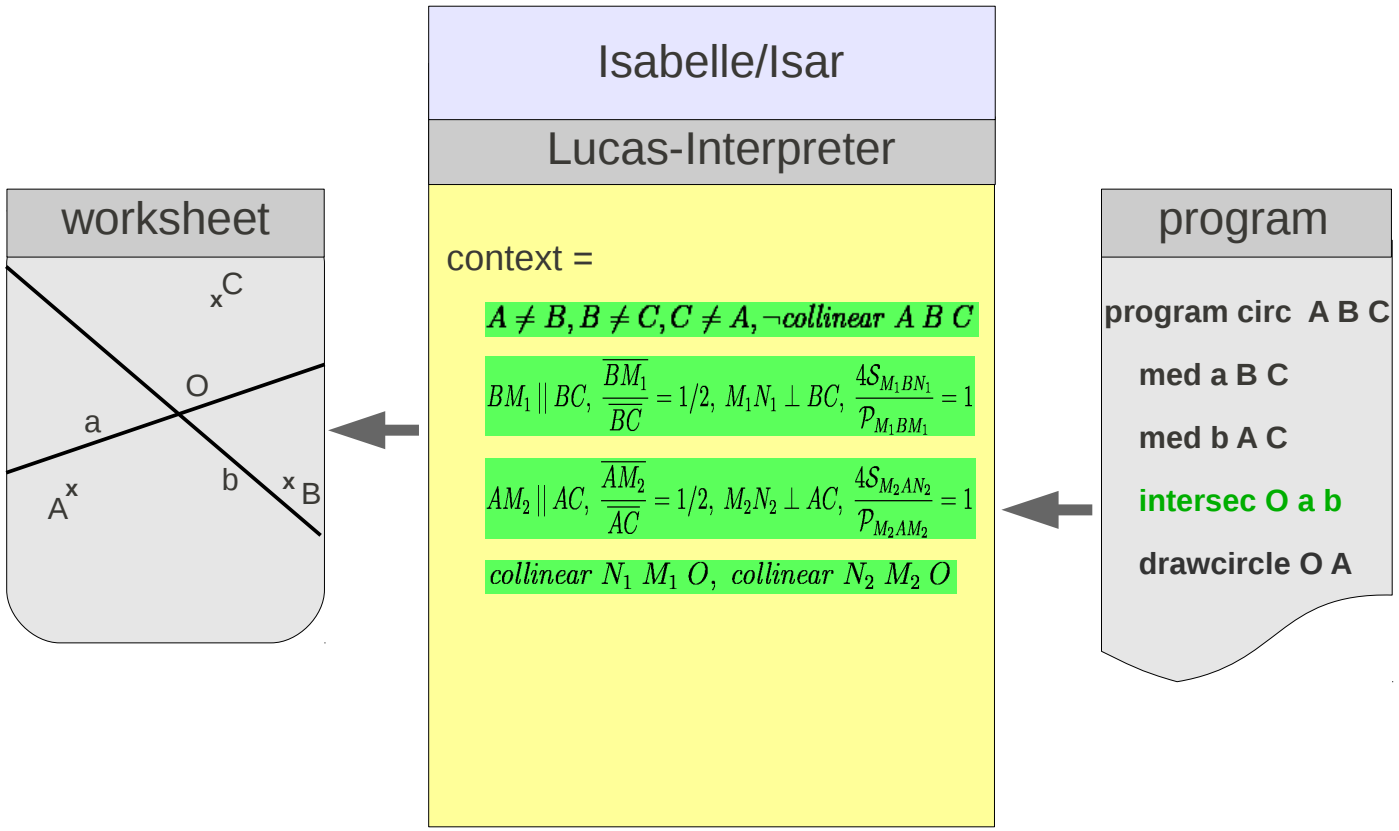
Breakpoint 1: context extended with area method formulas



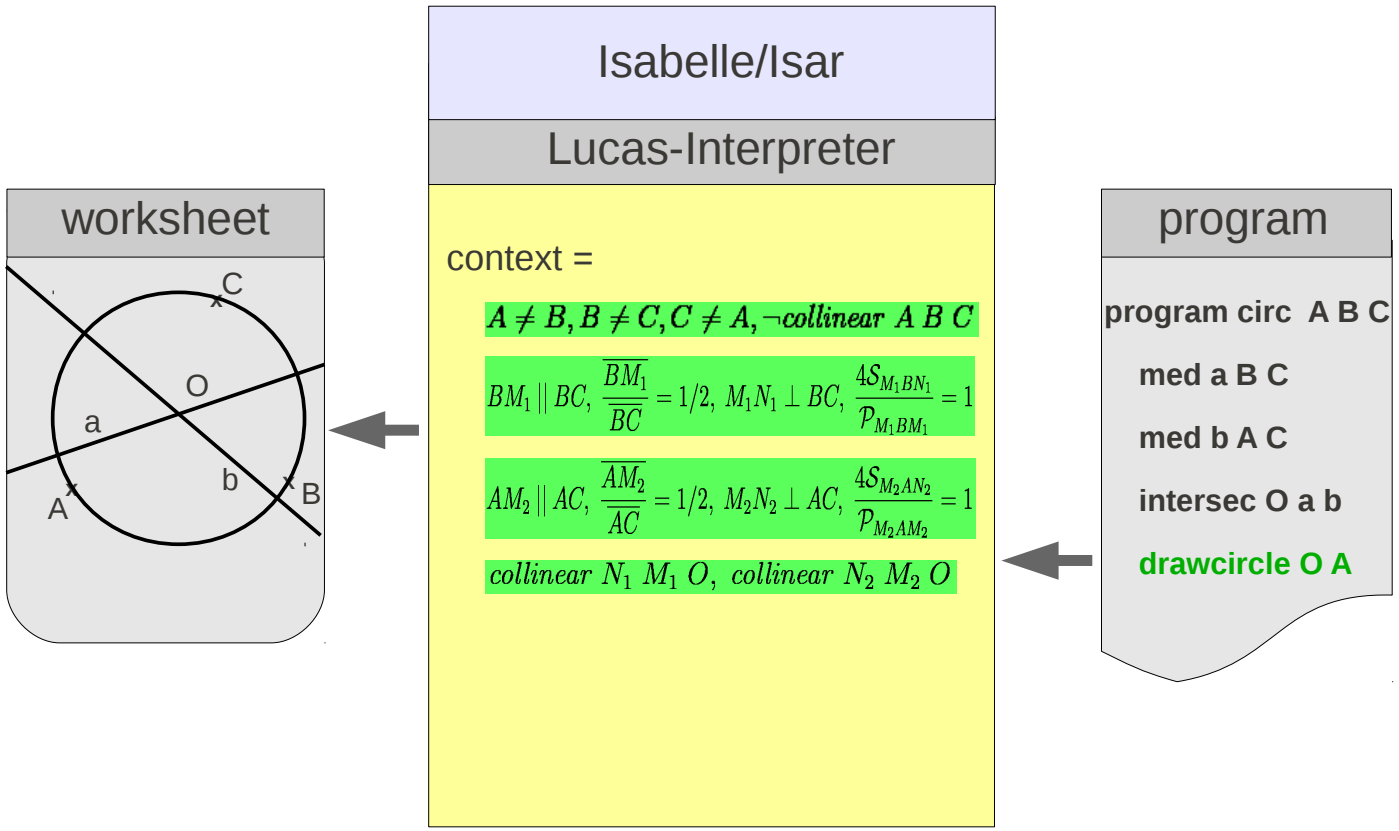
Breakpoint 2: context extended with area method formulas



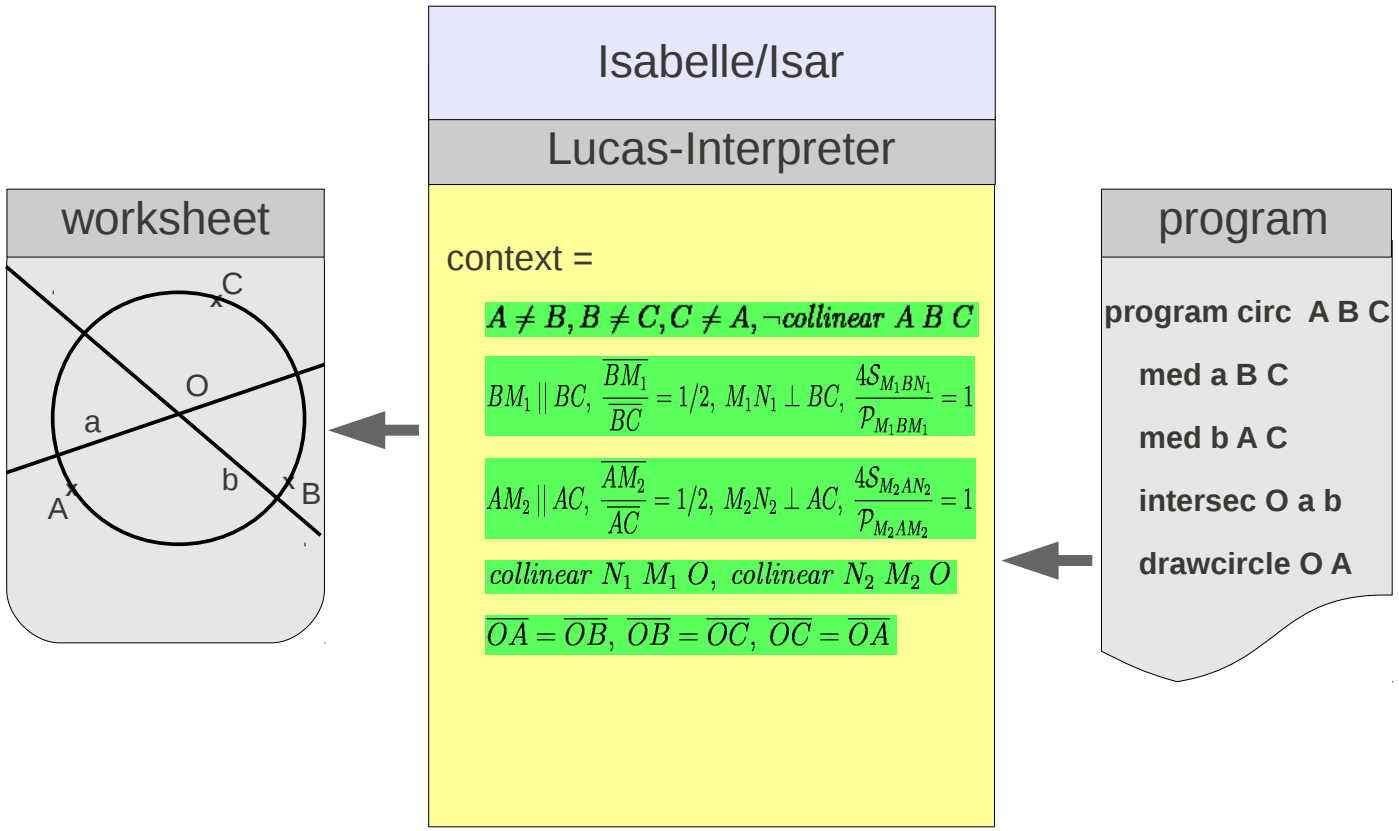
Breakpoint 3: context extended with area method formulas



Breakpoint 4: same context



Context provides data for proof of postcondition



A Lucas-Interpreter provides . . .

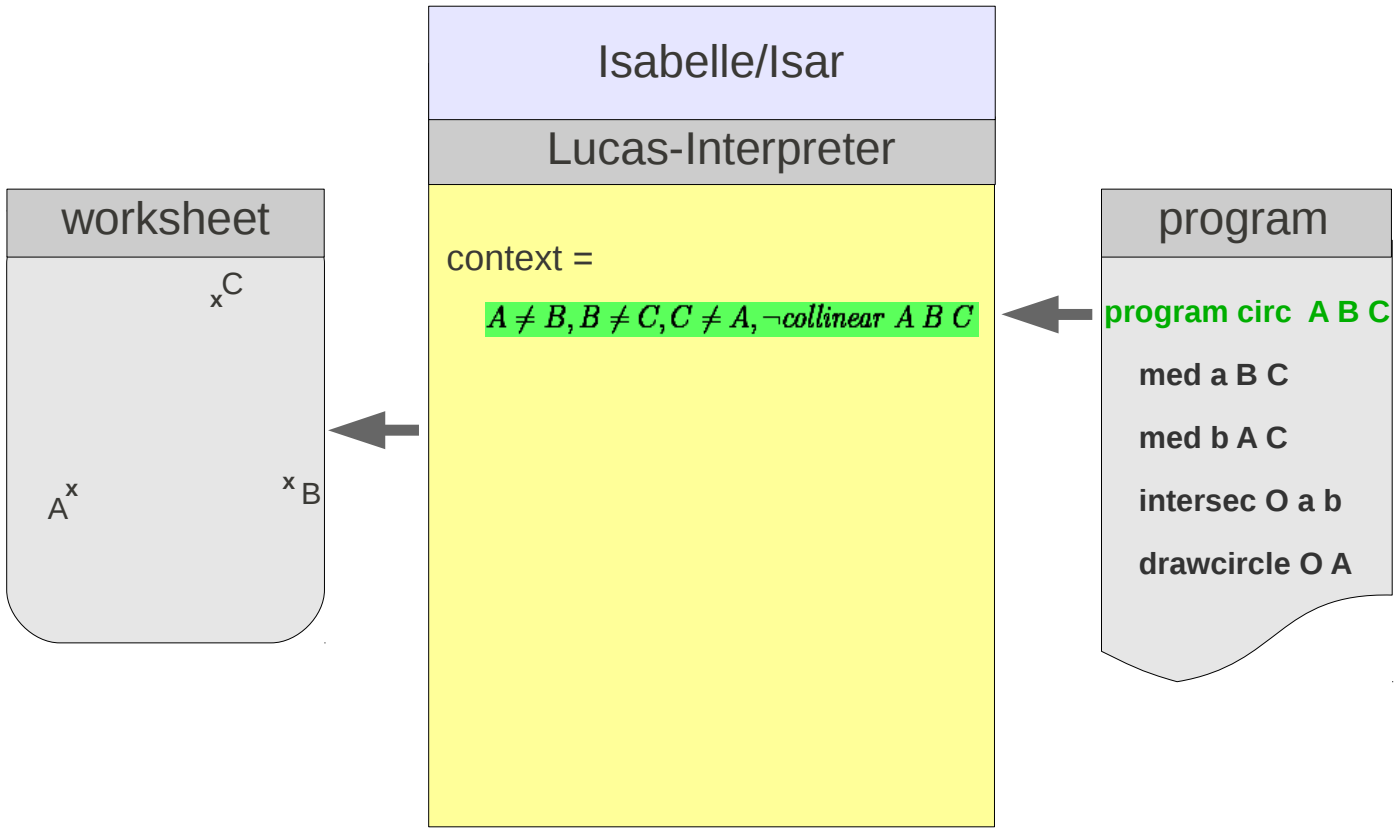
- **logical context** for specific program statements
- a final **context for proving** the postcondition
-
-
-
-
-
-

A Lucas-Interpreter provides . . .

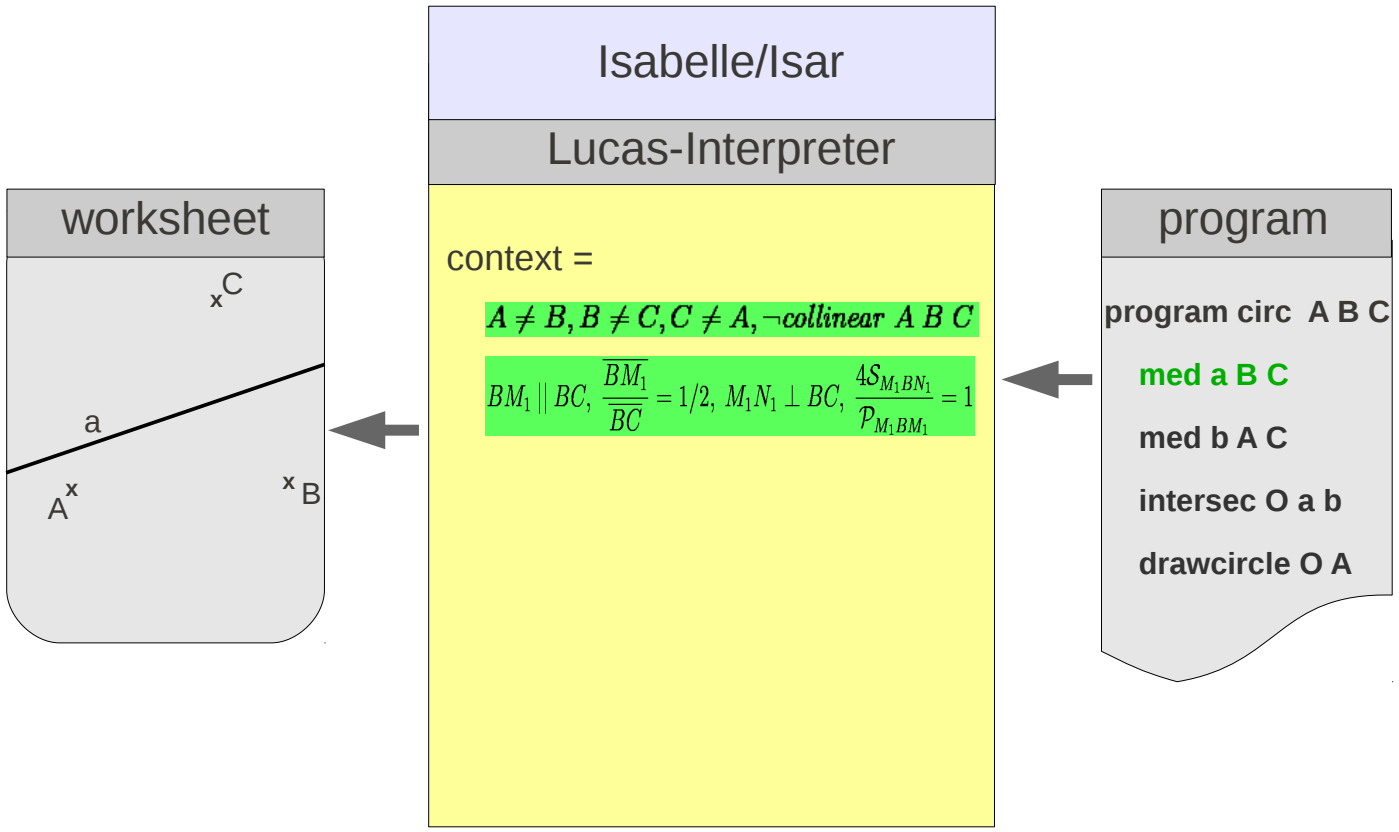
- **logical context** for specific program statements
- a final **context for proving** the postcondition
-
-
  -
  -
  -
  -
  -

Outline

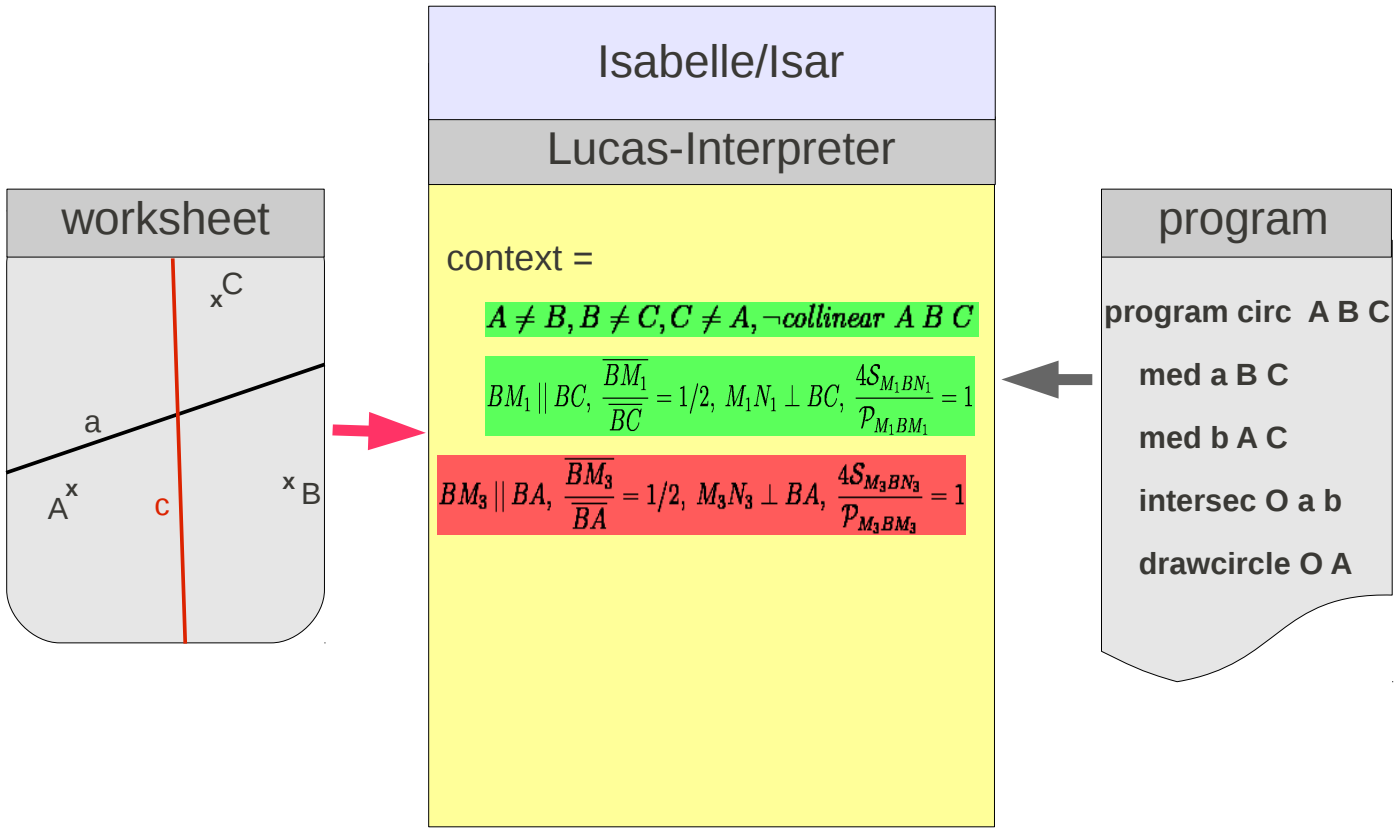
Somewhere during stepwise construction . . .



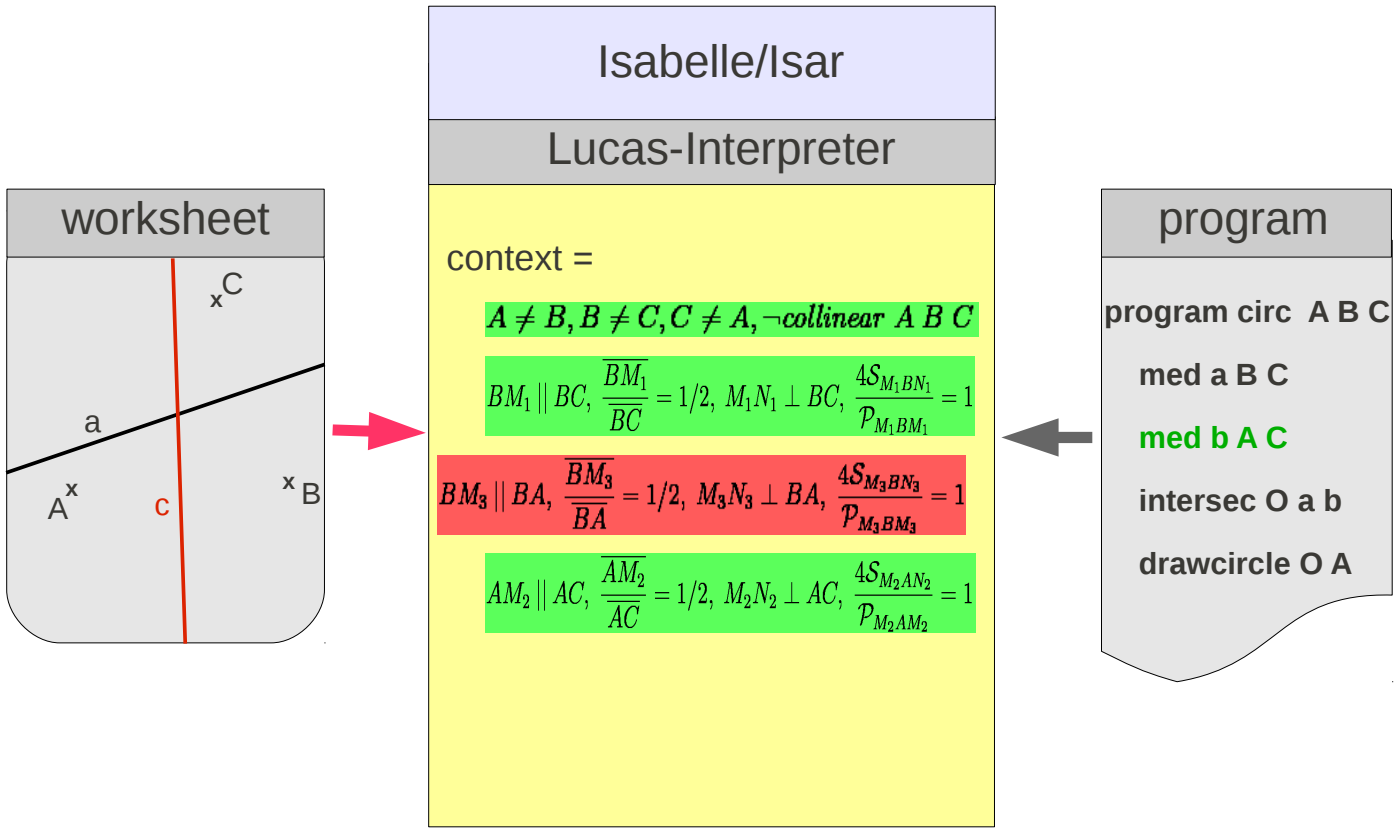
Somewhere during stepwise construction . . .



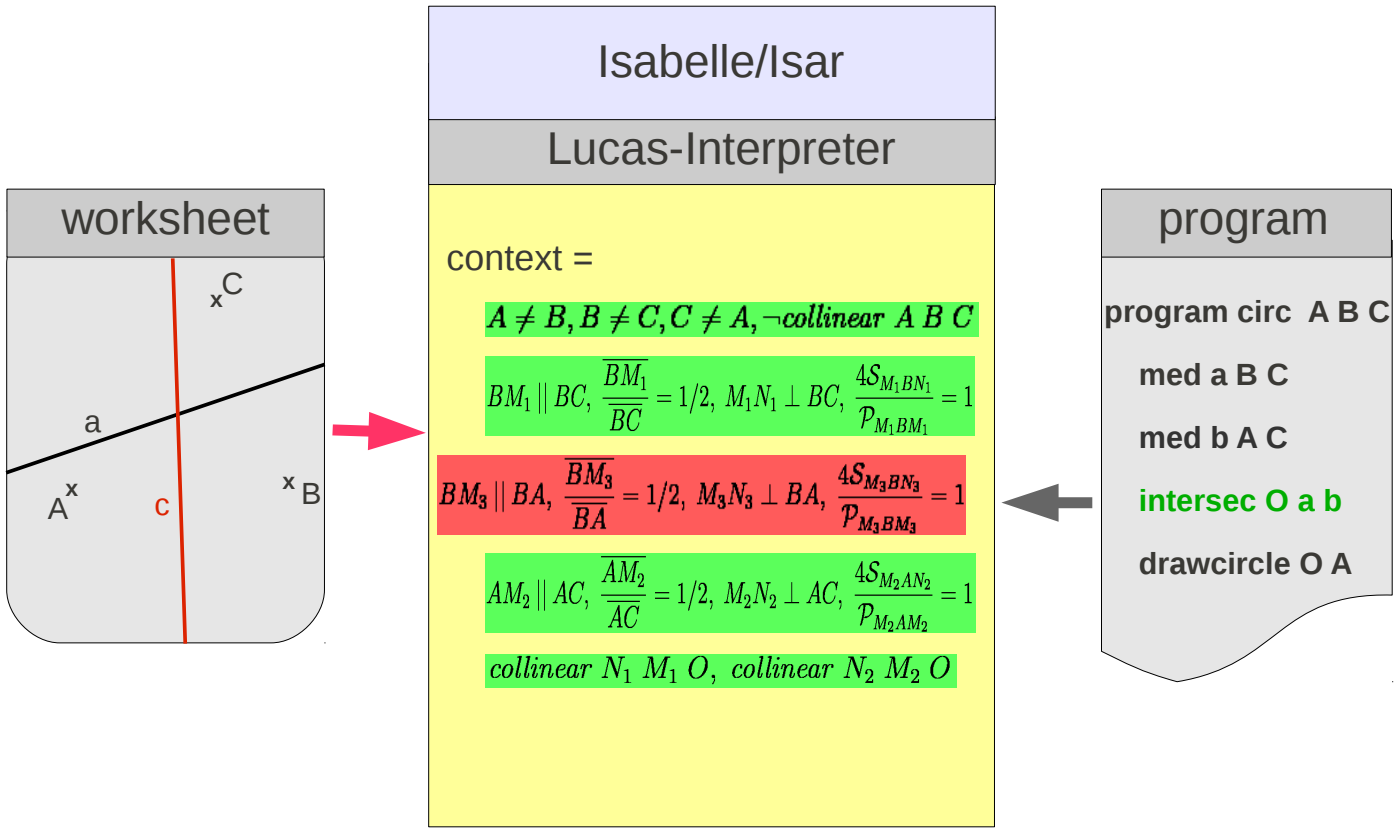
. . . the user inputs the next step: logical data created !



Lucas-Interpretation compares with subsequent contexts



Lucas-Interpretation compares with subsequent contexts



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 $\Longleftarrow$!!!
  - *interpolants* measure 'distance' to postcondition (?)
  - . . .

Outline

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` . . . . $BM_1 \parallel BC$, $\frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2}$, $M_1N_1 \perp BC$, $\frac{4S_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1$
   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 ?
   `med a B C` . . . . $BM_1 \parallel BC$, $\frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2}$, $M_1N_1 \perp BC$, $\frac{4S_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1$
   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 ?
   `med a B C` . . . . $BM_1 \parallel BC$, $\frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2}$, $M_1N_1 \perp BC$, $\frac{4S_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1$
   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 ?
   med a B C . . . . $BM_1 \parallel BC$, $\frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2}$, $M_1N_1 \perp BC$, $\frac{4S_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1$
   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 ?
   `med a B C` . . . . $BM_1 \parallel BC$, $\frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2}$, $M_1N_1 \perp BC$, $\frac{4S_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1$
   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 ?
   `med a B C` . . . . $BM_1 \parallel BC$, $\frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2}$, $M_1N_1 \perp BC$, $\frac{4\mathcal{S}_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1$
   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 ?
   `med a B C` . . . . $BM_1 \parallel BC$, $\frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2}$, $M_1N_1 \perp BC$, $\frac{4\mathcal{S}_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1$
   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 ?
   `med a B C` . . . . $BM_1 \parallel BC$, $\frac{\overline{BM_1}}{\overline{BC}} = \frac{1}{2}$, $M_1N_1 \perp BC$, $\frac{4\mathcal{S}_{M_1BN_1}}{\mathcal{P}_{M_1BM_1}} = 1$
   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:

`http://www.uc.pt/en/congressos/thedu`