User Guidance from Program Interpretation
Experiences from ISAC-experiments with Isabelle’09

Walther Neuper
Institute for Softwaretechnology
Graz University of Technology

Workshop on Automated Deduction and Geogebra
Feb.8, 2010, Castro Urdiales
1. Remarks on tutoring
   Status quo
   Requirements

2. Demo ISAC & Isabelle
   Technologies
   Architecture

3. Example 'circumcenter'
   Program code
   Specifications
   Contexts
   Work in progress

4. Summary
1. Remarks on tutoring
   Status quo
   Requirements

2. Demo ISAC & Isabelle
   Technologies
   Architecture

3. Example 'circumcenter'
   Program code
   Specifications
   Contexts
   Work in progress

4. Summary
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
   Status quo
   Requirements

2 Demo \textit{ISAC} & Isabelle
   Technologies
   Architecture

3 Example 'circumcenter'
   Program code
   Specifications
   Contexts
   Work in progress

4 Summary
Requirements for tutoring

The general requirements are basically

1. **check user input** as generous and liberal as possible

2. **guide the user** step by step towards a solution,
   guide towards the completion of a geom. construction

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

Demonstration of the 3 requirements with **algebra** in **ISAC**.
Requirements for tutoring

The general requirements are basically

1. check user input as generous and liberal as possible
2. guide the user step by step towards a solution,
   guide towards the completion of a geom. construction
3. explain steps on request by the user.

Demonstration of the 3 requirements with algebra in ISAC.
Requirements for tutoring

The general requirements are basically

1. **check user input** as generous and liberal as possible

2. **guide the user** step by step towards a solution,
   guide towards the completion of a geom. construction

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

Demonstration of the 3 requirements with algebra in ISAC.
Requirements for tutoring

The general requirements are basically

1. **check user input** as generous and liberal as possible

2. **guide the user** step by step towards a solution,
   guide towards the completion of a geom. construction

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

Demonstration of the 3 requirements with **algebra** in *ISAC*. 
Requirements for tutoring

The general requirements are basically

1. **check user input** as generous and liberal as possible

2. **guide the user** step by step towards a solution, guide towards the completion of a geom. construction

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

Demonstration of the 3 requirements with **algebra** in **ISAC**.
Remarks on tutoring
Status quo
Requirements

Demo ISAC & Isabelle
Technologies
Architecture

Example 'circumcenter'
Program code
Specifications
Contexts
Work in progress

Summary
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. simplifiers, SAT, SMT, area method etc.

2. **guide the user** step by step towards a solution, guide towards the completion of a geom. construction by interpretation in **debug-mode** with breakpoints at each step.

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

Tutoring and authoring both build upon points (1)–(3). Point (2) is in the focus subsequently.
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. simplifiers, SAT, SMT, area method etc.

2. guide the user step by step towards a solution, guide towards the completion of a geom. construction by interpretation in debug-mode with breakpoints at each step.

3. explain steps on request by the user by human-readable knowledge on a separate language layer.

Tutoring and authoring both build upon points (1)..(3). Point (2) is in the focus subsequently.
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. simplifiers, SAT, SMT, area method etc

2. **guide the user** step by step towards a solution, guide towards the completion of a geom. construction by **interpretation in debug-mode** with breakpoints at each step

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

Tutoring and authoring both build upon points (1)–(3). Point (2) is in the focus subsequently.
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. simplifiers, SAT, SMT, area method etc

2. guide the user step by step towards a solution, guide towards the completion of a geom. construction by interpretation in debug-mode with breakpoints at each step

3. explain steps on request by the user by human-readable knowledge on a separate language layer.

Tutoring and authoring both build upon points (1)..(3). Point (2) is in the focus subsequently.
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. simplifiers, SAT, SMT, area method etc.

2. **guide the user** step by step towards a solution, guide towards the completion of a geom. construction by **interpretation in debug-mode** with breakpoints at each step.

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

**Tutoring and authoring** both build upon points (1)–(3). Point (2) is in the focus subsequently.
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. simplifiers, SAT, SMT, area method etc

2. guide the user step by step towards a solution, guide towards the completion of a geom. construction by interpretation in debug-mode with breakpoints at each step

3. explain steps on request by the user by human-readable knowledge on a separate language layer.

Tutoring and authoring both build upon points (1)–(3). Point (2) is in the focus subsequently.
Remarks on tutoring
Status quo
Requirements

Demo ISAC & Isabelle Technologies
Architecture

Example 'circumcenter'
Program code
Specifications
Contexts
Work in progress

Summary
**ISAC's present experimental version**

![Diagram showing ISAC's architecture](image)

- **JavaSwing**
- **Isabelle**
  - Provers
  - Theories
  - Rewriting
  - Matching
  - Pretty printing

**Remarks on tutoring**
- Status quo
- Requirements

**Demo ISAC & Isabelle**
- Technologies
- Architecture

**Example 'circumcenter'**
- Program code
- Specifications
- Contexts
- Work in progress

**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?
Remarks on tutoring
Status quo
Requirements

Demo ISAC & Isabelle
Technologies
Architecture

Example 'circumcenter'
Program code
Specifications
Contexts
Work in progress

Summary
Program describing the construction

```
1  Script circumcenter A B C O =
2     let (P₁, P₂) = Take ε(P₁, P₂). P₁ ∈ {A, B, C} ∧ P₂ ∈ {A, B, C}
3           ∧ P₁ ≠ P₂
4       m₁ = Subproblem(Triangles, [geom, euclid, plane, median],
5                        [geom, euclid, plane, 2circ]) P₁ P₂ m₁
6     (P₃, P₄) = Take ε(P₃, P₄). P₃ ∈ {A, B, C} − {P₁, P₂} ∧
7          P₄ ∈ {A, B, C} ∧ P₃ ≠ P₄
8       m₂ = Subproblem(Triangles, [geom, euclid, plane, median],
9                        [geom, euclid, plane, 2circ]) P₃ P₄ m₂
10    in O = Take εO. m₁ ∩ m₂
```
Outline

1. Remarks on tutoring
   Status quo
   Requirements

2. Demo ISAC & Isabelle
   Technologies
   Architecture

3. Example 'circumcenter'
   Program code
   Specifications
   Contexts
   Work in progress

4. Summary
Specifications in the example

Specification of problem *circumcenter*

\[
\begin{align*}
in & : A, B, C \\
pre & : \neg \text{collinear } A B C \\
out & : O \\
post & : \epsilon O. \overline{OA} = \overline{OB} = \overline{OC}
\end{align*}
\]

Specification of (sub-)problem *median*

\[
\begin{align*}
in & : P, Q \\
pre & : P \neq Q \\
out & : m \\
post & : \epsilon m. m = \{ X. \overline{XP} = \overline{XQ} \}
\end{align*}
\]

Identifiers are typed: capital letters are points, lowercase are lines.
Pre : \(\neg\text{collinear } A B C\)

1. **Script circumcenter** \(A\ B\ C\ O =\)

2. \(\text{let } (P_1, P_2) = \text{Take } \varepsilon(P_1, P_2).\ P_1 \in \{A, B, C\} \land P_2 \in \{A, B, C\}\)

3. \(\land P_1 \neq P_2\)

Pre : \(P_1 \neq P_2\)

4. \(m_1 = \text{Subproblem}(\text{Triangles}, [\text{geom, euclid, plane, median}],\)

5. \([\text{geom, euclid, plane, 2circ}])\ P_1\ P_2\ m_1\)

Post : \(\varepsilon m_1.\ m_1 = \{P. PP_1 = PP_2\}\)

6. \((P_3, P_4) = \text{Take } \varepsilon(P_3, P_4).\ P_3 \in \{A, B, C\} \setminus \{P_1, P_2\}\land\)

7. \(P_4 \in \{A, B, C\} \land P_3 \neq P_4\)

Pre : \(P_3 \neq P_4\)

8. \(m_2 = \text{Subproblem}(\text{Triangles}, [\text{geom, euclid, plane, median}],\)

9. \([\text{geom, euclid, plane, 2circ}])\ P_3\ P_4\ m_2\)

Post : \(\varepsilon m_2.\ m_2 = \{P. PP_3 = PP_4\}\)

10. \(in\ O = \text{Take } \varepsilon O.\ m_1 \cap m_2\)

Post : \(\varepsilon O.\ \overline{OA} = \overline{OB} = \overline{OC}\)
Context during execution

Isar Contexts gather environment, assertions and others.

1  \textit{circumcenter} X Y Z M \\
   \{(A, X), (B, Y), (C, Z), (O, M), \neg \text{collinear } A B C\}

2,3 \textit{select} C B, \textit{for instance} \\
   \{(\ldots, (P_1, Z), (P_2, Y), P_1 \neq P_2\}

4,5 \textit{m}_1 = \textit{Subproblem} (\textit{Triangles}, \ldots \\
   \{(\ldots, (m_1, m_1), m_1 = \{P. \overline{PP_1} = \overline{PP_2}\}

6,7 \textit{select} A B, \textit{for instance} \\
   \{(\ldots, (P_3, X), (P_4, Y), P_3 \neq P_4\}

8,9 \textit{m}_2 = \textit{Subproblem} (\textit{Triangles}, \ldots \\
   \{(\ldots, (m_2, m_2), m_2 = \{P. \overline{PP_3} = \overline{PP_4}\}

10 \textit{O} = \textit{Take} \varepsilon O. m_1 \cap m_2 \\
   \{(\ldots, (O, M), \overline{OA} = \overline{OB} = \overline{OC}\)
Work in progress

- Adapt *TSAC's* program language to Isabelle’09
  - integrate Isar Contexts into the language
  - integrate Isabelle's provers, e.g. for lines 2,3 and 6,7
  - investigate “proof of satisfiability” for the postcondition w.r.t. a program
  - ...

- ? Adapt *TSAC's* program language to geometry ?
  - How relate dependency graphs in DGS to Contexts ?
  - Clarify the interface between representation (e.g. coordinates) and abstract construction (e.g. \( \{P. \overline{PP}_3 = \overline{PP}_4\} \))
  - Generation of explanations for steps seems different from algebra (more related to Context than to theorems applied).
  - ...

- ? Standard ML instead of Isabelle/HOL?
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.

A window of innovation seems open presently!
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.

A window of innovation seems open presently!
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.

A window of innovation seems open presently!