Program Languages with CTP Features?
On ISAC-experiments with Isabelle’09

Walther Neuper

Institute for Softwaretechnology
Graz University of Technology

Workshop on Formal and Automated Theorem Proving
January 2010, Beograd
Outline

1. Issues from e-learning
   - Idea
   - CTP — tutoring
   - ISAC tutor demonstration

2. CTP-based languages?
   - ISAC’s language
   - Language design generalized?

3. Convergent architecture
   - Isabelle history
   - ISAC joins Isabelle

4. Summary
1. Issues from e-learning
   Idea
   CTP — tutoring
   ISAC tutor demonstration

2. CTP-based languages?
   ISAC’s language
   Language design generalized?

3. Convergent architecture
   Isabelle history
   ISAC joins Isabelle

4. Summary
Design a program language for applied mathematics . . .

Design a language analogous to CAS-based languages but based on Computer Theorem Proving (CTP) such that programs implementing applied math automatically create tutoring on that math stuff. . . . such that tutoring becomes a side effect of ordinary math programs.
Design a program language for applied mathematics . . .

Design a language analogous to CAS-based languages but based on Computer Theorem Proving (CTP) such that programs implementing applied math automatically create tutoring on that math stuff.

. . . such that tutoring becomes a side effect of ordinary math programs.
Design a program language for applied mathematics . . .

Design a language analogous to CAS-based languages but based on Computer Theorem Proving (CTP) such that programs implementing applied math automatically create tutoring on that math stuff. . . such that tutoring becomes a side effect of ordinary math programs.
Design a program language for applied mathematics . . .

Design a language analogous to CAS-based languages but based on Computer Theorem Proving (CTP) such that programs implementing applied math automatically create tutoring on that math stuff.

. . . such that tutoring becomes a side effect of ordinary math programs.
Design a program language for applied mathematics...

Design a language analogous to CAS-based languages but based on Computer Theorem Proving (CTP) such that programs implementing applied math automatically create tutoring on that math stuff.

... such that tutoring becomes a side effect of ordinary math programs.
1. Issues from e-learning
   Idea
   CTP — tutoring
   ISAC tutor demonstration

2. CTP-based languages?
   ISAC’s language
   Language design generalized?

3. Convergent architecture
   Isabelle history
   ISAC joins Isabelle

4. Summary
Requirements in tutoring applied math

A tutoring system for applied math serves by . . .

1. checking user-input “correct modulo a theory”
2. providing surveys on subproblems and specifications
3. guiding the user step-wise towards a solution

Demonstration of experiments with the ISAC tutor
A tutoring system for applied math serves by . . .

1. checking user-input “correct modulo a theory”
2. providing surveys on subproblems and specifications
3. guiding the user step-wise towards a solution

Demonstration of experiments with the \texttt{ISAC} tutor
Requirements in tutoring applied math

A tutoring system for applied math serves by . . .

1. checking user-input “correct modulo a theory”
2. providing surveys on subproblems and specifications
3. guiding the user step-wise towards a solution

Demonstration of experiments with the ISAC tutor
A tutoring system for applied math serves by . . .

1. checking user-input “correct modulo a theory”
2. providing surveys on subproblems and specifications
3. guiding the user step-wise towards a solution

Demonstration of experiments with the *ISAC* tutor
A tutoring system for applied math serves by . . .

1. checking user-input “correct modulo a theory"
2. providing surveys on subproblems and specifications
3. guiding the user step-wise towards a solution

Demonstration of experiments with the ISAC tutor
Outline

1. Issues from e-learning
   - Idea
   - CTP — tutoring
   - ISAC tutor demonstration

2. CTP-based languages?
   - ISAC’s language
   - Language design generalized?

3. Convergent architecture
   - Isabelle history
   - ISAC joins Isabelle

4. Summary
Resume of the demonstration

The ISAC tutor serves with . . .

1. checking user-input “correct modulo a theory” by use of Isabelle provers (e.g. simplifier): CTP !

2. providing surveys on subproblems and specifications by use of Isabelle contexts (e.g. pre-conditions): CTP !

3. guiding the user step-wise towards a solution using a single-stepping interpreter: program language !

If CTP is involved, what about program languages ?
Resume of the demonstration

The ISAC tutor serves with . . .

1 checking user-input “correct modulo a theory” by use of Isabelle provers (e.g. simplifier): CTP!

2 providing surveys on subproblems and specifications by use of Isabelle contexts (e.g. pre-conditions): CTP!

3 guiding the user step-wise towards a solution using a single-stepping interpreter: program language!

If CTP is involved, what about program languages?
Resume of the demonstration

The *ISAC* tutor serves with . . .

1. checking user-input “correct modulo a theory”  
   *by use of Isabelle provers (e.g. simplifier): CTP!*

2. providing surveys on subproblems and specifications  
   *by use of Isabelle contexts (e.g. pre-conditions): CTP!*

3. guiding the user step-wise towards a solution  
   *using a single-stepping interpreter: program language!*

If CTP is involved, what about program languages?
Resume of the demonstration

The \textit{ISAC} tutor serves with . . .

1. checking user-input “correct modulo a theory” 
   \textit{by use of Isabelle provers (e.g. simplifier): CTP !}

2. providing surveys on subproblems and specifications
   \textit{by use of Isabelle contexts (e.g. pre-conditions): CTP !}

3. guiding the user step-wise towards a solution
   \textit{using a single-stepping interpreter: program language !}

If CTP is involved, what about program languages ?
Resume of the demonstration

The ISAC tutor serves with . . .

1. checking user-input “correct modulo a theory” by use of Isabelle provers (e.g. simplifier): CTP !
2. providing surveys on subproblems and specifications by use of Isabelle contexts (e.g. pre-conditions): CTP !
3. guiding the user step-wise towards a solution using a single-stepping interpreter: program language !

If CTP is involved, what about program languages ?’
Resume of the demonstration

The ISAC tutor serves with . . .

1. checking user-input “correct modulo a theory” by use of Isabelle provers (e.g. simplifier): CTP!
2. providing surveys on subproblems and specifications by use of Isabelle contexts (e.g. pre-conditions): CTP!
3. guiding the user step-wise towards a solution using a single-stepping interpreter: program language!

If CTP is involved, what about program languages?’
Issues from e-learning

Idea
CTP — tutoring
ISAC tutor demonstration

CTP-based languages?

ISAC's language
Language design generalized?

Convergent architecture

Isabelle history
ISAC joins Isabelle

Summary
Resume of the demonstration

\textit{ISAC}'s experimental program language . . .

1. is purely functional, \newline user-in/output handled by interpreter; \newline \textit{programming} math in a typed, functional language!

2. checks specifications of subproblems, \newline interactive specification is invoked by interpreter; \newline \textit{programming}: pre-conditions guard method invocation!

3. maintains contexts (predicates, type-constraints) \newline which assists in checking user-input; \newline \textit{programming}: logic checks in runtime improve safety!

Which further advantages from CTP for programming ???
Resume of the demonstration

\(\text{\textsc{Isac}}\)'s experimental program language . . .

1. **is purely functional,**  
   \textit{user-in/output handled by interpreter;}  
   \textit{programming math in a typed, functional language !}

2. **checks specifications of subproblems,**  
   \textit{interactive specification is invoked by interpreter;}  
   \textit{programming: pre-conditions guard method invocation !}

3. **maintains contexts (predicates, type-constraints)**  
   \textit{which assists in checking user-input;}  
   \textit{programming: logic checks in runtime improve safety !}

Which further advantages from CTP for programming ??
Resume of the demonstration

\(\text{ISAC}'s\) experimental program language . . .

1. **is purely functional,**
   
   *user-in/output handled by interpreter; programming math in a typed, functional language!*

2. **checks specifications of subproblems,**
   
   *interactive specification is invoked by interpreter; programming: pre-conditions guard method invocation!*

3. **maintains contexts (predicates, type-constraints)**
   
   *which assists in checking user-input; programming: logic checks in runtime improve safety!*

Which further advantages from CTP for programming ???
Resume of the demonstration

\( \text{ISAC}'s \) experimental program language …

1. **is purely functional,**  
   *user-in/output handled by interpreter;*  
   *programming math in a typed, functional language!*

2. **checks specifications of subproblems,**  
   *interactive specification is invoked by interpreter;*  
   *programming: pre-conditions guard method invocation!*

3. **maintains contexts (predicates, type-constraints)**  
   *which assists in checking user-input;*  
   *programming: logic checks in runtime improve safety!*

Which further advantages from CTP for programming ???
Resume of the demonstration

**ISAC**'s experimental program language . . .

1. **is purely functional,**
   
   *user-in/output handled by interpreter;*  
   
   *programming* *math in a typed, functional language!*

2. **checks specifications of subproblems,**
   
   *interactive specification is invoked by interpreter;*  
   
   *programming: pre-conditions guard method invocation!*

3. **maintains contexts (predicates, type-constraints)**
   
   *which assists in checking user-input;*  
   
   *programming: logic checks in runtime improve safety!*

Which further advantages from CTP for programming ???
Resume of the demonstration

"ISAC"s experimental program language . . .

1. is purely functional, 
   *user-in/output handled by interpreter*;
   *programming math in a typed, functional language!*

2. checks specifications of subproblems, 
   *interactive specification is invoked by interpreter*;
   *programming: pre-conditions guard method invocation!*

3. maintains contexts (*predicates, type-constraints*)
   *which assists in checking user-input*;
   *programming: logic checks in runtime improve safety!*

Which further advantages from CTP for programming ???
Resume of the demonstration

\( \text{ISAC}'s \) experimental program language . . .

1. is purely functional,
   \( \text{user-in/output handled by interpreter;} \)
   \( \text{programming math in a typed, functional language!} \)

2. checks specifications of subproblems,
   \( \text{interactive specification is invoked by interpreter;} \)
   \( \text{programming: pre-conditions guard method invocation!} \)

3. maintains contexts (predicates, type-constraints)
   \( \text{which assists in checking user-input;} \)
   \( \text{programming: logic checks in runtime improve safety!} \)

Which further advantages from CTP for programming ???
Programming with CTP?

Walther Neuper

Issues from e-learning
- Idea
- CTP — tutoring
- ISAC tutor demonstration

CTP-based languages?
- ISAC’s language
- Language design generalized?

Convergent architecture
- Isabelle history
- ISAC joins Isabelle

Summary

Outline

1. Issues from e-learning
   - Idea
   - CTP — tutoring
   - ISAC tutor demonstration

2. CTP-based languages?
   - ISAC’s language
   - Language design generalized?

3. Convergent architecture
   - Isabelle history
   - ISAC joins Isabelle

4. Summary
Features for CTP-based languages?

A CTP-based language for (applied) math, which . . .

1. . . is purely functional
2. . . checks specifications of subproblems
3. . . maintains contexts (predicates, type-constraints)
4. . . organizes knowledge local to theories, contexts
5. . . supports proof of correctness of programs
6. . . supports local pretty printing (\LaTeX, MathML)

. . . is this interesting for (appl.) math programmers?
To which other developments user guidance can hook up?
Features for CTP-based languages?

A CTP-based language for (applied) math, which . . .

1. . . . is purely functional
2. . . . checks specifications of subproblems
3. . . . maintains contexts (predicates, type-constraints)
4. . . . organizes knowledge local to theories, contexts
5. . . . supports proof of correctness of programs
6. . . . supports local pretty printing ($\LaTeX$, MathML)
7. . . .

. . . is this interesting for (appl.) math programmers?
To which other developments user guidance can hook up?
Features for CTP-based languages?

A CTP-based language for (applied) math, which . . .

1. . . . is purely functional
2. . . . checks specifications of subproblems
3. . . . maintains contexts (predicates, type-constraints)
4. . . . organizes knowledge local to theories, contexts
5. . . . supports proof of correctness of programs
6. . . . supports local pretty printing (\LaTeX, MathML)
7. . .

. . . is this interesting for (appl.) math programmers?
To which other developments user guidance can hook up?
Features for CTP-based languages?

A CTP-based language for (applied) math, which . . .

1. . . is purely functional
2. . . checks specifications of subproblems
3. . . maintains contexts (predicates, type-constraints)
4. . . organizes knowledge local to theories, contexts
5. . . supports proof of correctness of programs
6. . . supports local pretty printing (\LaTeX, MathML)
7. . .

. . . is this interesting for (appl.) math programmers?
To which other developments user guidance can hook up?
Features for CTP-based languages?

A CTP-based language for (applied) math, which . . .

1. . . . is purely functional
2. . . . checks specifications of subproblems
3. . . . maintains contexts (predicates, type-constraints)
4. . . . organizes knowledge local to theories, contexts
5. . . . supports proof of correctness of programs
6. . . . supports local pretty printing (\LaTeX, MathML)

. . .

. . . is this interesting for (appl.) math programmers?
To which other developments user guidance can hook up?
Features for CTP-based languages?

A CTP-based language for (applied) math, which . . .

1. . . is purely functional
2. . . checks specifications of subproblems
3. . . maintains contexts (predicates, type-constraints)
4. . . organizes knowledge local to theories, contexts
5. . . supports proof of correctness of programs
6. . . supports local pretty printing (\LaTeX, MathML)

. . . is this interesting for (appl.) math programmers?
To which other developments user guidance can hook up?
Features for CTP-based languages?

A CTP-based language for (applied) math, which . . .

1. . . is purely functional
2. . . checks specifications of subproblems
3. . . maintains contexts (predicates, type-constraints)
4. . . organizes knowledge local to theories, contexts
5. . . supports proof of correctness of programs
6. . . supports local pretty printing (\LaTeX, MathML)
7. . .

. . . is this interesting for (appl.) math programmers? To which other developments user guidance can hook up?
Features for CTP-based languages?

A CTP-based language for (applied) math, which . . .

1. . . . is purely functional
2. . . . checks specifications of subproblems
3. . . . maintains contexts (predicates, type-constraints)
4. . . . organizes knowledge local to theories, contexts
5. . . . supports proof of correctness of programs
6. . . . supports local pretty printing (\LaTeX, MathML)
7. . . .

. . . is this interesting for (appl.) math programmers?
To which other developments user guidance can hook up?
Issues from e-learning

CTP — tutoring

ISAC tutor demonstration

CTP-based languages?

ISAC’s language

Language design generalized?

Convergent architecture

Isabelle history

ISAC joins Isabelle

Summary
The original Isabelle architecture

ML toplevel

Isabelle

Provers
Theories
rewriting
matching
pretty printing

Emacs

Summary

Convergent architecture

Isabelle history

ISAC joins

Isabelle

CTP-based languages?

ISAC’s language

Language design

genralized?

Issues from e-learning

Idea

CTP — tutoring

ISAC tutor
demonstration

Programming with CTP?

Walther Neuper
Isar proof language hides goal/subgoal mechanism

Emacs ProofGeneral

<table>
<thead>
<tr>
<th>Isar toplevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isabelle</td>
</tr>
<tr>
<td>Provers</td>
</tr>
<tr>
<td>Theories</td>
</tr>
<tr>
<td>rewriting</td>
</tr>
<tr>
<td>matching</td>
</tr>
<tr>
<td>pretty printing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Isar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contexts</td>
</tr>
</tbody>
</table>

Summary
Scala will enhance interoperability for GUIs etc

CTP-based languages?
- ISAC's language
- Language design generalized?

Convergent architecture
- ISAC joins Isabelle

Summary

Programming with CTP?
- Walther Neuper

Issues from e-learning
- Idea
- CTP — tutoring
- ISAC tutor demonstration

Editor (jEdit, Lyx)
Parser
Programming with CTP?

Walther Neuper

Issues from e-learning
- Idea
- CTP — tutoring
- ISAC tutor demonstration

CTP-based languages?
- ISAC’s language
- Language design generalized?

Convergent architecture
- Isabelle history
- ISAC joins Isabelle

Summary
Present ISAC adds Scripts and interpreter.
The next version will exploit Isars contexts.
\textit{ISAC} adds programs (Isabelle terms, “Scripts”) and interpreter
Could there be Standard ML instead the Isabelle terms?
Could there be Standard ML instead the Isabelle terms?
Then the same program could be production code !?!
Then the same program could be production code !?!
Then the same program could be production code !?!

Math production ....

fun appliedmath(args) val somevar = ... result ...
Towards CTP-based languages?

To features of CAS-based languages . . .

- typed matching and rewriting

. . . adding CTP-based ones towards a language, which . . .

? is purely functional

? checks specifications of subproblems

? maintains contexts (predicates, type-constraints)

? organizes knowledge local to theories, contexts

? supports proof of correctness of programs

? supports local pretty printing (\LaTeX, MathML)

? etc . . .

. . . is this interesting for (appl.) math programmers ?

How fit into the PolyML-Scala-Isabelle development ?
Towards CTP-based languages?

To features of CAS-based languages . . .

- typed matching and rewriting

. . . adding CTP-based ones towards a language, which . . .

- is purely functional
- checks specifications of subproblems
- maintains contexts (predicates, type-constraints)
- organizes knowledge local to theories, contexts
- supports proof of correctness of programs
- supports local pretty printing ($\LaTeX$, MathML)
- etc . . .

. . . is this interesting for (appl.) math programmers ?
How fit into the PolyML-Scala-Isabelle development ?
Towards CTP-based languages?

To features of CAS-based languages . . .

- typed matching and rewriting

. . . adding CTP-based ones towards a language, which . . .

- is purely functional
- checks specifications of subproblems
- maintains contexts (predicates, type-constraints)
- organizes knowledge local to theories, contexts
- supports proof of correctness of programs
- supports local pretty printing ($\LaTeX$, MathML)
- etc ???

. . . is this interesting for (appl.) math programmers?
How fit into the PolyML-Scala-Isabelle development?
Towards CTP-based languages?

To features of CAS-based languages . . .

• typed matching and rewriting

. . . adding CTP-based ones towards a language, which . . .

? is purely functional

? checks specifications of subproblems

? maintains contexts (predicates, type-constraints)

? organizes knowledge local to theories, contexts

? supports proof of correctness of programs

? supports local pretty printing (\LaTeX, MathML)

? etc ???

. . . is this interesting for (appl.) math programmers ?

How fit into the PolyML-Scala-Isabelle development ?
Towards CTP-based languages?

To features of CAS-based languages . . .

- typed matching and rewriting

. . . adding CTP-based ones towards a language, which . . .

? is purely functional

? checks specifications of subproblems

? maintains contexts (predicates, type-constraints)

? organizes knowledge local to theories, contexts

? supports proof of correctness of programs

? supports local pretty printing ($\LaTeX$, MathML)

? etc ???

. . . is this interesting for (appl.) math programmers?

How fit into the PolyML-Scala-Isabelle development?
Towards CTP-based languages?

To features of CAS-based languages . . .

- typed matching and rewriting

. . . adding CTP-based ones towards a language, which . . .

? is purely functional

? checks specifications of subproblems

? maintains contexts (predicates, type-constraints)

? organizes knowledge local to theories, contexts

? supports proof of correctness of programs

? supports local pretty printing ($\LaTeX$, MathML)

? etc ???

. . . is this interesting for (appl.) math programmers ?
How fit into the PolyML-Scala-Isabelle development ?