Program Languages with CTP Features ?

On $\mathcal{ISAC}$-experiments with Isabelle'09

Walther Neuper

Institute for Softwaretechnology
Graz University of Technology

Workshop on Formal and Automated Theorem Proving
January 2010, Beograd

Outline

Outline

Design a program language for applied mathematics . . .

Design a language analoguous 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 analoguous 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 analoguous 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 analoguous 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 analoguous 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.

Outline

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 $\mathcal{ISAC}$ 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 $\mathcal{ISAC}$ 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 $\mathcal{ISAC}$ 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 $\mathcal{ISAC}$ 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 $\mathcal{ISAC}$ tutor

Outline

Resume of the demonstration

The $\mathcal{ISAC}$ 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 $\mathcal{ISAC}$ 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 $\mathcal{ISAC}$ 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 $\mathcal{ISAC}$ 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 $\mathcal{ISAC}$ 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 $\mathcal{ISAC}$ 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 ?'

Outline

Resume of the demonstration

$\mathcal{ISAC}$'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. **maintans 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

$\mathcal{ISAC}$'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. **maintans 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

$\mathcal{ISAC}$'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. **maintans 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

$\mathcal{ISAC}$'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. **maintans 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. **maintans 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

$\mathcal{ISAC}$'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. **maintans 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

$\mathcal{ISAC}$'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. **maintans contexts (predicates, type-constraints)**
   which assists in checking user-input;
   programming: logic checks in runtime improve safety !

Which further advantages from CTP for programming ???

Outline

Features for CTP-based languages ?

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

1. . . . is purely functional
2. . . . checks specifications of subproblems
3. . . . maintans 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. . . . maintans 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. . . . maintans 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. . . . maintans 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. . . . maintans 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. . . . maintans 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. ... maintans 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 ?

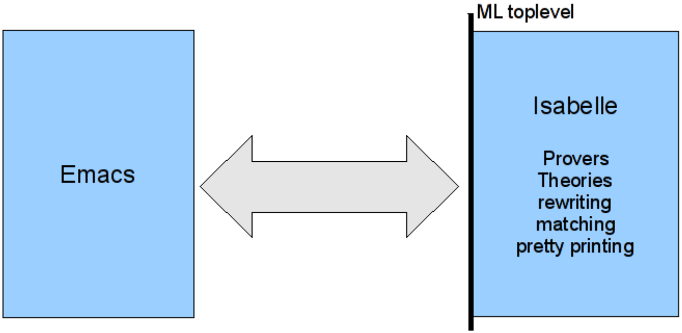
A CTP-based language for (applied) math, which …

1. … is purely functional
2. … checks specifications of subproblems
3. … maintans 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. **…**

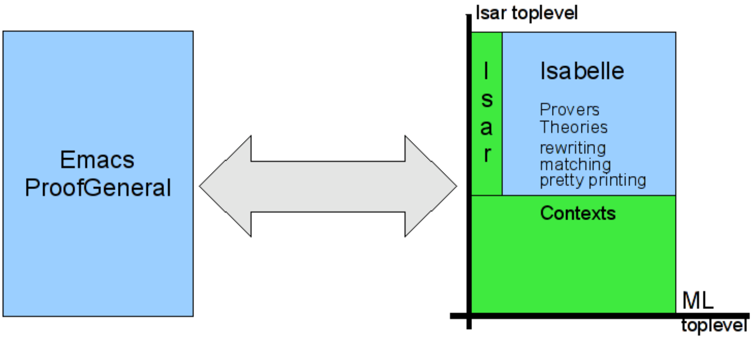
… is this interesting for (appl.) math programmers ?
To which other developments user guidance can hook up ?

Outline

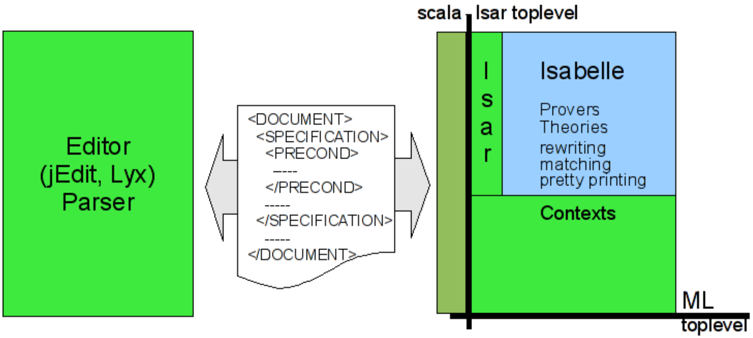
The original Isabelle architecture



Isar proof language hides goal/subgoal mechanism

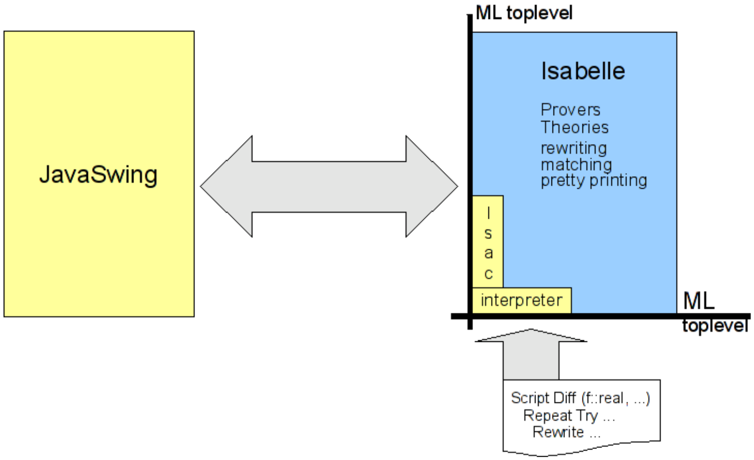


Scala will enhance interoperability for GUIs etc

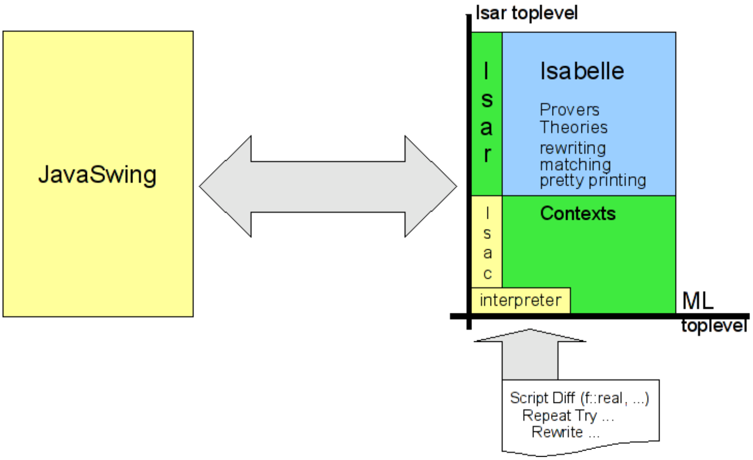


Outline

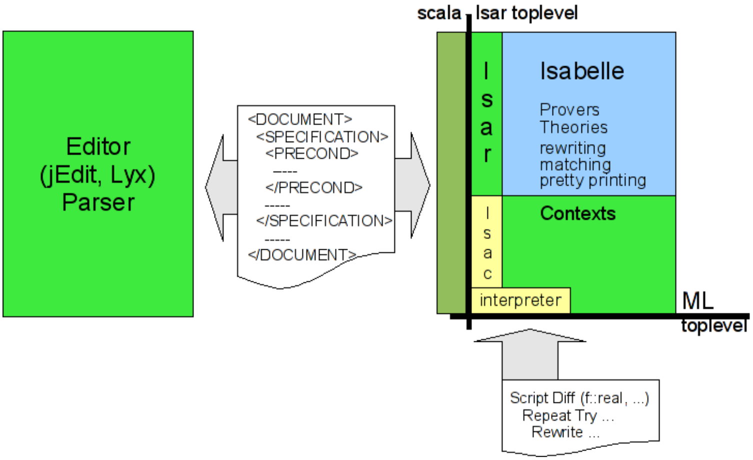
Present $\mathcal{ISAC}$ adds Scripts and interpreter



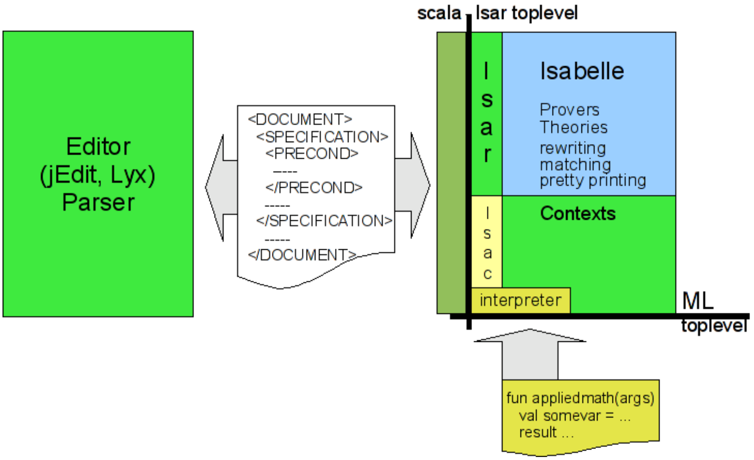
The next version will exploit Isars contexts



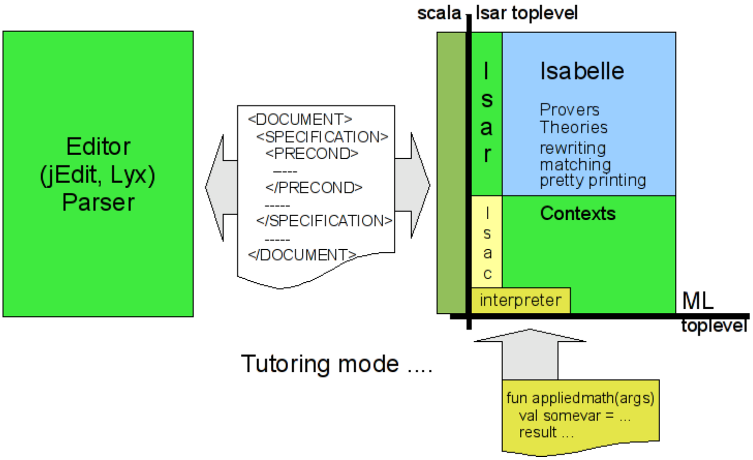
$\mathcal{ISAC}$ adds programs (Isabelle terms, “Scripts”) and interpreter



Could there be Standard ML instead the Isabelle terms ?

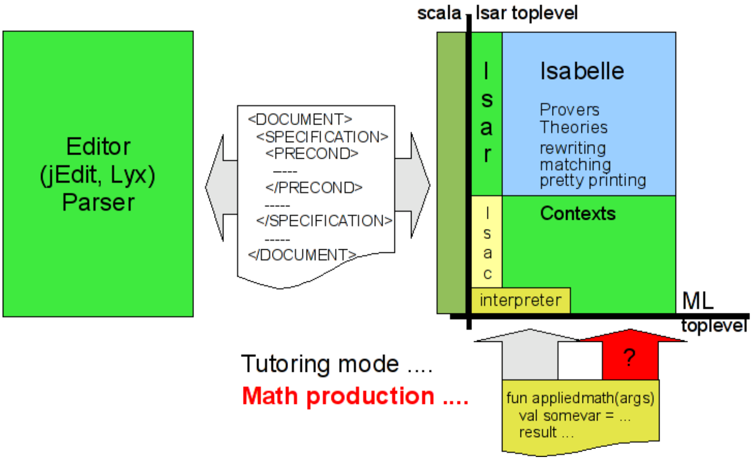


Could there be Standard ML instead the Isabelle terms ?



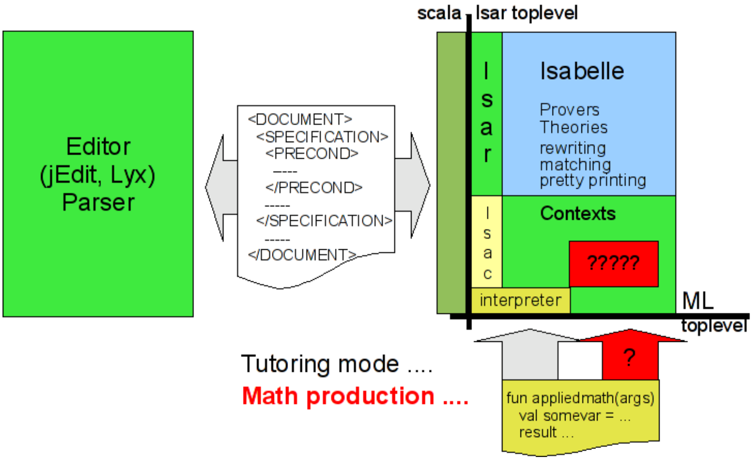
Tutoring mode ....

Then the same program could be production code !?!



```
fun appliedmath(args)
  val somevar = ...
  result ...
```

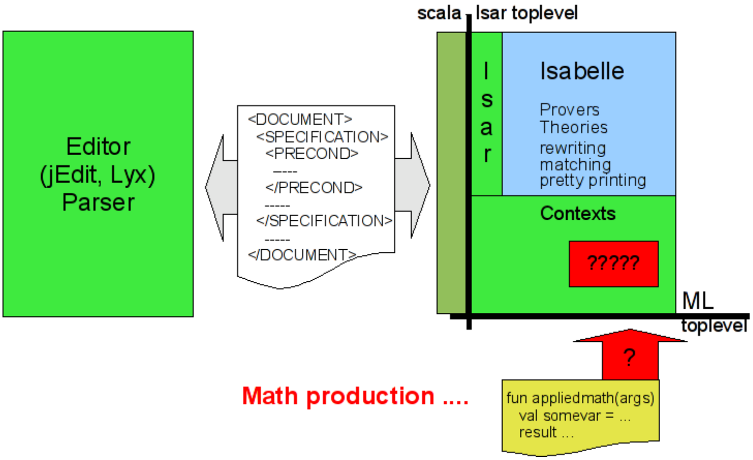
Then the same program could be production code !?!



Tutoring mode ....

Math production ....

Then the same program could be production code !?!



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

? maintans 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
? maintans 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

? maintans 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
- ? maintans 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

? maintans 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

? maintans 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 ?