Mechanised Justification in "Systems that Explain Themselves" for Mathematics Education

Walther Neuper

IICM, Institute for Computer Media, University of Technology.
Graz, Austria

eduTPS: Working Group on Justification in Doing Math
at CADGME, Coimbra, Portugal
June 27, 2018
Outline

1. “Systems that Explain Themselves”?

2. Mechanical Explanation and Language Layers
   - Term Language
   - Proof Language
   - Specification Language
   - “Next step guidance”
   - Programming Language

3. Conclusions
“Systems that Explain Themselves”?

- Systems for proofs: well-known theorem provers (TP), e.g. Coq, PVS, HOL, Isabelle have
  - math knowledge deduced from first principles (axioms)
  - so, elements of math **knowledge** “explain themselves”
  - **usage** for proof does **not** explain itself

  → short demo of Isabelle

- Systems for engineering mathematics: only prototypes, e.g. 4ferries (by R.J. Back), Mathtoys, *ISAC*
  - need to build upon TPs (justifications !)
  - need to step-wise construct problem solutions
  - need to support modularisation into sub-problems
  - ... such that **usage** and **knowledge** “explain themselves”

  → Isabelle/*ISAC* will serve as example
“Systems that Explain Themselves”? 

- Systems for proofs: well-known theorem provers (TP), e.g. Coq, PVS, HOL, Isabelle have 
  - math knowledge deduced from first principles (axioms) 
  - so, elements of math knowledge “explain themselves”
  - usage for proof does not explain itself

→ short demo of Isabelle

- Systems for engineering mathematics: only prototypes, e.g. 4ferries (by R.J. Back), Mathtoys, ISAC
  - need to build upon TPs (justifications !)
  - need to step-wise construct problem solutions
  - need to support modularisation into sub-problems
  - . . . such that usage and knowledge “explain themselves”

→ Isabelle/ISAC will serve as example
“Systems that Explain Themselves”? 

- Systems for proofs: well-known theorem provers (TP), e.g. Coq, PVS, HOL, Isabelle have
  - math knowledge deduced from first principles (axioms)
  - so, elements of math **knowledge** “explain themselves”
  - **usage** for proof does **not** explain itself

  → short demo of Isabelle

- Systems for engineering mathematics: only prototypes, e.g. 4ferries (by R.J. Back), Mathtoys, ISAC
  - need to build upon TP (justifications !)
  - need to step-wise construct problem solutions
  - need to support modularisation into sub-problems
  - ... such that **usage** and **knowledge** “explain themselves”

  → Isabelle/ISAC will serve as example
“Systems that Explain Themselves”?  

- Systems for proofs: well-known theorem provers (TP), e.g. Coq, PVS, HOL, Isabelle have  
  - math knowledge deduced from first principles (axioms)  
  - so, elements of math **knowledge** “explain themselves”  
  - **usage** for proof does **not** explain itself  

→ short demo of Isabelle  

- Systems for engineering mathematics: only prototypes, e.g. 4ferries (by R.J. Back), Mathtoys, ISAC  
  - need to build upon TPs (justifications !)  
  - need to step-wise construct problem solutions  
  - need to support modularisation into sub-problems  
  - . . . such that **usage** and **knowledge** “explain themselves”  

→ Isabelle/ISAC will serve as example
“Systems that Explain Themselves”?

- Systems for proofs: well-known theorem provers (TP), e.g. Coq, PVS, HOL, Isabelle have
  - math knowledge deduced from first principles (axioms)
  - so, elements of math **knowledge** “explain themselves”
  - **usage** for proof does **not** explain itself

→ short demo of Isabelle

- Systems for engineering mathematics: only prototypes, e.g. 4ferries (by R.J. Back), Mathtoys, *ISAC*
  - need to build upon TPs (justifications !)
  - need to step-wise construct problem solutions
  - need to support modularisation into sub-problems
  - ... such that **usage** and **knowledge** “explain themselves”

→ Isabelle/*ISAC* will serve as example
“Systems that Explain Themselves”?

- Systems for proofs: well-known theorem provers (TP), e.g. Coq, PVS, HOL, Isabelle have
  - math knowledge deduced from first principles (axioms)
  - so, elements of math **knowledge** “explain themselves”
  - **usage** for proof does **not** explain itself

  → short demo of Isabelle

- Systems for engineering mathematics: only prototypes, e.g. 4ferries (by R.J. Back), Mathtoys, **ISAC**
  - need to build upon TPs (justifications !)
  - need to step-wise construct problem solutions
  - need to support modularisation into sub-problems
  - ... such that **usage** and **knowledge** “explain themselves”

  → Isabelle/**ISAC** will serve as example
“Systems that Explain Themselves”? 

- Systems for proofs: well-known theorem provers (TP), e.g. Coq, PVS, HOL, Isabelle have 
  - math knowledge deduced from first principles (axioms) 
  - so, elements of math **knowledge** “explain themselves” 
  - usage for proof does **not** explain itself 

  —→ short demo of Isabelle 

- Systems for engineering mathematics: only prototypes, e.g. 4ferries (by R.J. Back), Mathtoys, *ISAC* 
  - need to build upon TPs (justifications !) 
  - need to step-wise construct problem solutions 
  - need to support modularisation into sub-problems 
  - ... such that usage and knowledge “explain themselves” 

  —→ Isabelle/*ISAC* will serve as example
“Systems that Explain Themselves”?

• Systems for proofs: well-known theorem provers (TP), e.g. Coq, PVS, HOL, Isabelle have
  • math knowledge deduced from first principles (axioms)
  • so, elements of math **knowledge** “explain themselves”
  • **usage** for proof does **not** explain itself

→ short demo of Isabelle

• Systems for engineering mathematics: only prototypes, e.g. 4ferries (by R.J. Back), Mathtoys, **ISAC**
  • need to build upon TPs (justifications !)
  • need to step-wise construct problem solutions
  • need to support modularisation into sub-problems
  • ... such that **usage** and **knowledge** “explain themselves”

→ Isabelle/ISAC will serve as example
1. "Systems that Explain Themselves"?

2. Mechanical Explanation and Language Layers
   Term Language
   Proof Language
   Specification Language
   "Next step guidance"
   Programming Language

3. Conclusions
The demo has shown . . .

- principal benefits
  - uniformity over all domains of mathematics
  - type system efficiently excludes ambiguities
  - clear description of functions and respective rules

- added value of implementation
  - a formula’s elements are connected with definitions
  - types are transparent by mouse pointer
  - feedback to input of formulas
  - structure of formulas, i.e. sub-terms are transparent
  - internal representation adaptable to engineers’ needs
The demo has shown . . .

- principal benefits
  - uniformity over all domains of mathematics
  - type system efficiently excludes ambiguities
  - clear description of functions and respective rules

- added value of implementation
  - a formula’s elements are connected with definitions
  - types are transparent by mouse pointer
  - feedback to input of formulas
  - structure of formulas, i.e. sub-terms are transparent
  - internal representation adaptable to engineers’ needs
The demo has shown . . .

- **principal benefits**
  - uniformity over all domains of mathematics
  - type system efficiently excludes ambiguities
  - clear description of functions and respective rules

- **added value of implementation**
  - a formula’s elements are connected with definitions
  - types are transparent by mouse pointer
  - feedback to input of formulas
  - structure of formulas, i.e. sub-terms are transparent
  - internal representation adaptable to engineers’ needs
1. “Systems that Explain Themselves”?

2. Mechanical Explanation and Language Layers
   - Term Language
   - Proof Language
   - Specification Language
     - “Next step guidance”
   - Programming Language

3. Conclusions
Proof Language . . .

. . . adapts to conventions of engineering mathematics:

Figure: Conventional worksheet on ISAC’s front-end
Proof Language

- **Principal benefits**
  - calculations in a conventional format
  - all steps of calculation in a consistent framework
  - each step is justified by theorems
  - specific steps equivalent to Computer Algebra
  - Computer Algebra decomposed into elementary steps

- **Added value of implementation**
  - change from survey to detail in the calculation tree (collapsing and expanding)
  - justification for any step can be inspected on demand
  - steps can be redone while trying alternative ways
  - alternatives can be tried in parallel windows
Proof Language

• Principal benefits
  • calculations in a conventional format
  • all steps of calculation in a consistent framework
  • each step is justified by theorems
  • specific steps equivalent to Computer Algebra
  • Computer Algebra decomposed into elementary steps

• Added value of implementation
  • change from survey to detail in the calculation tree (collapsing and expanding)
  • justification for any step can be inspected on demand
  • steps can be redone while trying alternative ways
  • alternatives can be tried in parallel windows
“Systems that Explain Themselves”?

2. Mechanical Explanation and Language Layers
   Term Language
   Proof Language
   Specification Language
   “Next step guidance”
   Programming Language

3. Conclusions
Specification Language

Formal specification of the previous **Solution:**

01 Problem (Biegelinie, [Biegelinien])
02 Specification:
03    Model:
04      Given : Traegerlaenge L, Streckenlast q_0
05      Where : q_0 ist_integrierbar_auf [0, L]
06      Find   : Biegelinie y
07      Relate : Randbedingungen [Q 0 = q_0 \cdot L, M_b L = 0, y 0 = 0, \frac{d}{dx} y 0 = 0]
08 References:
09      Theory : Biegelinie
10    x Problem : ["Biegelinien"]
11      o Method : ["IntegrierenUndKonstanteBestimmen2"]
12  Solution:

Hidden data for “next step guidance”:

[ ( [ Traegerlaenge L, Streckenlast q_0, Biegelinie y,
    Randbedingungen [ Q 0 = q_0 \cdot L, M_b L = 0, y 0 = 0, \frac{d}{dx} y 0 = 0], FunktionsVariable x ]
  ("Biegelinie", ["Biegelinien"], ["IntegrierenUndKonstanteBestimmen2"] ) ) ]
Specification Language

- **Principal benefits**
  - formal specification prepares mechanical solution
  - pre-condition determines solvability
  - post-condition makes essence of a problem explicit
  - problems decomposed into sub-problems (with specifications)

- **Added value of implementation**
  - specifications can be easily searched and tried
  - trees of specifications allow automated refinement
  - successfully specified problems solved by key stroke
  - sub-problems can be interactively arranged
  - specifications as black boxes raises abstraction in problem solving, see slide movie
Specification Language

- **Principal benefits**
  - formal specification prepares mechanical solution
  - pre-condition determines solvability
  - post-condition makes essence of a problem explicit
  - problems decomposed into sub-problems (with specifications)

- **Added value of implementation**
  - specifications can be easily searched and tried
  - trees of specifications allow automated refinement
  - successfully specified problems solved by key stroke
  - sub-problems can be interactively arranged
  - specifications as black boxes raises abstraction in problem solving, see slide movie
Start Example

Problem: From a horizontally lying pipe with a diameter of 8 cm there are 5 liters of water flowing out per second. At what height is this pipe, if the horizontal distance between outlet and incidence on the floor is 80 cm?

Note: First determine the exit velocity (by use of the volume of water per second and of the cross-section area.)
From a horizontally lying pipe with a diameter of 8 cm there are 5 liters of water flowing out per second. At what height is this pipe, if the horizontal distance between outlet and incidence on the floor is 80 cm?

Note: First determine the exit velocity (by use of the volume of water per second and of the cross-section area.)
Problem modelled ok

Model:

Given: Diameter \( d = 8 \text{ cm} \), FlowRate \( \phi = 54/\text{s} \),
HorizontalDistance \( s = 80 \text{ cm} \)
Where: \( d > 0 \wedge \phi > 0 \wedge s > 0 \)
Find: HeightOfPipe \( h \)

From a horizontally lying pipe with a diameter of 8 cm there are 5 liters of water flowing out per second. At what height is this pipe, if the horizontal distance between outlet and incidence on the floor is 80 cm?

Note: First determine the exit velocity (by use of the volume of water per second and of the cross-section area.)
Start considering sub-problems

Model:
Given: Diameter $d = 8\text{ cm}$, FlowRate $\phi = 54\text{ l/s}$,
HorizontalDistance $s = 80\text{ cm}$
Where: $d > 0 \land \phi > 0 \land s > 0$
Find: RightOfPipe $h$

Find: $h\text{ m}$

Problem [area-of-circle]

$\phi \text{l/s}$, $s\text{ cm}$
Select sub-problems
Select sub-problems

Model:

Given: Diameter \( d = 8 \text{ cm} \), FlowRate \( \phi = 54 \text{l/s} \),
HorizontalDistance \( s = 80 \text{ cm} \),
Where: \( d > 0 \land \phi > 0 \land s > 0 \),
Find: RightOfPipe \( h \)

Problem [rational, equation]

\[
d \text{ cm, } \phi \text{ l/s, } s \text{ cm}
\]

Problem [velocity-space-time, find-time]

\[
\nu = \frac{s}{t}
\]

Find: \( h \) \text{ m}
Select sub-problems

Problem [rational, equation]
\[
\text{Problem [velocity-space-time, find-time]}
\]
\[
v = \frac{s}{t}
\]
Problem [flow-rate, find-velocity]
\[
v = \frac{\phi}{A_{\text{circle}}}
\]
Find: \( h \) m
Select sub-problems

Problem [rational, equation]

Problem [velocity-space-time, find-time]

Problem [flow-rate, find-velocity]

Problem [free-fall]

Find: h m
Delete irrelevant sub-problems

Problem \([\text{rational, equation}]\)

Problem \([\text{flow-rate, find-velocity}]\)

Problem \([\text{velocity-space-time, find-time}]\)

\[ \nu = \frac{s}{t} \]

Problem \([\text{flow-rate, find-velocity}]\)

\[ \nu = \frac{\phi}{A_{\text{circle}}} \]

Problem \([\text{free-fall}]\)

\[ h = \frac{g}{2} \cdot t^2 \]

Find: \(h\) m
Select relevant sub-problems

Problem [area-of-circle]
\[ A_{\text{circle}} = \left(\frac{d}{2}\right)^2 \cdot \pi \]

Problem [velocity-space-time, find-time]
\[ v = \frac{s}{t} \]

Problem [flow-rate, find-velocity]
\[ v = \frac{\phi}{A_{\text{circle}}} \]

Problem [free-fall]
\[ h = \frac{g}{2} \cdot t^2 \]

Given:
- Diameter \( d = 8 \text{ cm} \)
- Flow Rate \( \phi = 54 \text{ l/s} \)
- Horizontal Distance \( s = 80 \text{ cm} \)

Where:
- \( d > 0 \)
- \( \phi > 0 \)
- \( s > 0 \)

Find:
- \( h \text{ m} \)
What is given / has to be found?

- $d$ cm, $\varphi$ l/s, $s$ cm
- Problem [area-of-circle]: $A_{circle} = \left(\frac{d}{2}\right)^2 \cdot \pi$
- Problem [velocity-space-time, find-time]: $v = \frac{s}{t}$
- Problem [flow-rate, find-velocity]: $v = \frac{\varphi}{A_{circle}}$
- Problem [free-fall]: $h = \frac{g}{2} \cdot t^2$

Find: $h$ m
What is given / has to be found?

Problem [flow-rate, find-velocity]

\[ \nu = \frac{\phi}{A_{\text{circle}}} \]

Problem [velocity-space-time, find-time]

\[ \nu = \frac{s}{t} \]

Problem [free-fall]

\[ h = \frac{g}{2} \cdot t^2 \]
Connect “Given” and “Find”

Problem [flow-rate, find-velocity]

\[ \nu = \phi \]

Problem [velocity-space-time, find-time]

\[ \nu = \frac{s}{t} \]

Problem [free-fall]

\[ h = \frac{g}{2} \cdot t^2 \]

Find: \( h \) m
Connect “Given” and “Find”

Problem [area-of-circle]
\[ A_{\text{circle}} = \left( \frac{d}{2} \right)^2 \cdot \pi \]

Problem [velocity-space-time, find-time]
\[ v = \frac{s}{t} \]

Problem [flow-rate, find-velocity]
\[ v = \frac{\phi}{A_{\text{circle}}} \]

Problem [free-fall]
\[ h = \frac{g}{2} \cdot t^2 \]

Find: \( h \) m
Connect “Given” and “Find”

Problem [flow-rate, find-velocity]

\[ \nu = \frac{\phi}{A_{\text{circle}}} \]

Problem [velocity-space-time, find-time]

\[ \nu = \frac{s}{t} \]

Problem [free-fall]

\[ h = \frac{g}{2} \cdot t^2 \]

Find: \( h \) m
Connect “Given” and “Find”

Problem [flow-rate, find-velocity]

Given: Diameter $d = 8 \text{ cm}$, FlowRate $\phi = 54 \text{ l/s}$,
HorizontalDistance $s = 80 \text{ cm}$

Find: HeightOfPipe $h$

Problem [velocity-space-time, find-time]

Problem [free-fall]

Find: $h \text{ m}$

Problem [area-of-circle]

Problem [velocity-space-time, find-time]

Problem [flow-rate, find-velocity]

Problem [free-fall]
Connect “Given” and “Find”

Problem [flow-rate, find-velocity]
\[ \nu = \phi \]

Problem [velocity-space-time, find-time]
\[ \nu = \frac{s}{t} \]

Problem [free-fall]
\[ h = \frac{g}{2} \cdot t^2 \]

Find: \( h \) m

\( d \) cm, \( \phi \) l/s, \( s \) cm

\( A \) cm², \( v \) m/s

\( t \) s

\( A \) m², \( \phi \) m³/s

\( v \) m/s

\( t \) s

\( A \) m

\( \phi \) m³/s

\( v \) m/s

\( t \) s
Dangling connection ???

Problem [flow-rate, find-velocity]

\[ \nu = \frac{\phi}{A} \]

Problem [velocity-space-time, find-time]

\[ \nu = \frac{s}{t} \]

Problem [area-of-circle]

\[ A_{\text{circle}} = \left(\frac{d}{2}\right)^2 \cdot \pi \]

Find: \( h \) m

Problem [free-fall]

\[ h = \frac{g}{2} \cdot t^2 \]
Rearrange sub-problems

Problem [area-of-circle]

\[ A_{\text{circle}} = \left(\frac{d}{2}\right)^2 \cdot \pi \]

Problem [velocity-space-time, find-time]

\[ v = \frac{s}{t} \]

Problem [flow-rate, find-velocity]

\[ v = \frac{\phi}{A_{\text{circle}}} \]

Problem [free-fall]

\[ h = \frac{g}{2} \cdot t^2 \]
Flipped two sub-problems

- Problem [area-of-circle]
  \[ A_{\text{circle}} = \left(\frac{d}{2}\right)^2 \cdot \pi \]
  Find: \( h \) m, \( h \) m

- Problem [flow-rate, find-velocity]
  \[ v = \frac{\phi}{A_{\text{circle}}} \]
  Find: \( v \) m/s

- Problem [velocity-space-time, find-time]
  \[ v = \frac{s}{t} \]
  \[ t \] s

- Problem [free-fall]
  \[ h = \frac{g}{2} \cdot t^2 \]

Given: Diameter \( d = 8 \) cm, FlowRate \( \phi = 54 \) l/s,
HorizontalDistance \( s = 80 \) cm
Where: \( d > 0 \times \phi > 0 \times s > 0 \)
Find: RightOfPipe \( h \)
Connect “Given” and “Find”
Connect “Given” and “Find”

Problem [area-of-circle]

Problem [flow-rate, find-velocity]

Problem [velocity-space-time, find-time]

Problem [free-fall]
Connect “Given” and “Find”

Problem [area-of-circle]

Problem [flow-rate, find-velocity]

Problem [velocity-space-time, find-time]

Problem [free-fall]

Find: \( h \ m \)

\[ A_{\text{circle}} = \left( \frac{d}{2} \right)^2 \cdot \pi \]

\[ v = \frac{\phi}{A_{\text{circle}}} \]

\[ \nu = \frac{\phi}{s} \]

\[ h = \frac{g}{2} \cdot t^2 \]
Connect “Given” and “Find”

**Problem [area-of-circle]**

\[ A_{\text{circle}} = \left( \frac{d}{2} \right)^2 \cdot \pi \]

**Find:** \( h \) \( m \)

**Problem [flow-rate, find-velocity]**

\[ v = \frac{\phi}{A_{\text{circle}}} \]

**Problem [velocity-space-time, find-time]**

\[ v = \frac{S}{t} \]

**Problem [free-fall]**

\[ h = \frac{g}{2} \cdot t^2 \]

**Find:** \( h \) \( m \)
Connect “Given” and “Find”

Find: \( h \ m \)

\( h = \frac{g}{2} \cdot t^2 \)

\( d \ cm, \ \varphi \ l/s, \ s \ cm \)

Problem [area-of-circle]

\[ A_{\text{circle}} = \left( \frac{d}{2} \right)^2 \cdot \pi \]

Problem [flow-rate, find-velocity]

\[ v = \frac{\varphi}{A_{\text{circle}}} \]

\[ v \ m/s \]

Problem [velocity-space-time, find-time]

\[ v = \frac{s}{t} \]

\[ t \ s \]

Problem [free-fall]

\[ h = \frac{g}{2} \cdot t^2 \]

\( h \ m \)
All connections finished
Care about unit conversions
Solution with units only

Given: Diameter $d = 8$ cm, FlowRate $\phi = 54 / s$,
      HorizontalDistance $s = 80$ cm

Where: $d > 0 \land \phi > 0 \land s > 0$
Find: RightOfPipe $h$

Solution:
Problem [area-of-circle]:

$$A_{\text{circle}} \text{ cm}$$

$$A_{\text{circle}} \text{ m}$$

$$\phi = 5 \frac{1}{s}$$

$$\phi = 0.005 \frac{m^3}{s}$$

Problem [flow-rate, find-velocity]:

$$v \frac{m}{s}$$

$$s = 80 \text{ cm}$$

$$s = 0.8 \text{ m}$$

Problem [velocity-space-time, find-time]:

$$t \frac{m}{s}$$

Problem [free-fall]:

$$h \text{ m}$$
Solution complete

Solution:

1. Problem \([\text{area-of-circle}]\)
   
   \[ A_{\text{circle}} = \frac{50}{cm^2} \]
   
   \[ A_{\text{circle}} = 0.005 \text{ m}^2 \]
   
   \[ \phi = 5 \frac{l}{s} \]
   
   \[ \phi = 0.005 \frac{m^3}{s} \]

2. Problem \([\text{flow-rate, find-velocity}]\)

\[ v = 1 \frac{m}{s} \]

3. Problem \([\text{velocity-space-time, find-time}]\)

\[ t = 0.8 \frac{m}{s} \]

4. Problem \([\text{free-fall}]\):

\[ h = 3.2 \text{ m} \]

Check postcond \([\text{composed, movement, no-6}]\)
“Systems that Explain Themselves”? 

Mechanical Explanation and Language Layers
Term Language
Proof Language
Specification Language
“Next step guidance”
Programming Language

Conclusions
“Next step guidance” . . .

... in specifying a problem:

If data for each variant for constructing a specification (one variant shown above) are given, then the system can guide the student in completing a specification.

... in step-wise constructing a solution:

If a program describes how to solve a problem defined by a formal specification, then this program run by Lucas-Interpretation
  - determines a next step (if requested by the student)
  - checks input of the student using the logical context.
“Next step guidance” . . .

• . . . in specifying a problem:

If data for each variant for constructing a specification (one variant shown above) are given, then the system can guide the student in completing a specification

• . . . in step-wise constructing a solution:

If a program describes how to solve a problem defined by a formal specification, then this program run by Lucas-Interpretation

• determines a next step (if requested by the student)
• checks input of the student using the logical context.
1 “Systems that Explain Themselves”?

2 Mechanical Explanation and Language Layers
   Term Language
   Proof Language
   Specification Language
   “Next step guidance”
   Programming Language

3 Conclusions
Programming Language . . .

. . . for authors of mathematics knowledge.

Isabelle provides a “function package” for programming.

Added value of this implementation:

- syntax errors are indicated accurately
- type annotations shift into the initial signature
- less type annotations are required
- syntax highlighting specific for constants etc
- free variables on right-hand-sides are rejected

Students might watch progress within a solution like in a debugger (on request).
Conclusions

TP technology provides mechanical explanations due to

- principal benefits
- added value of implementation
- and “next step guidance”

of various kinds on different language layers — all explanations come on users’ request!

Field tests will show, whether “systems that explain themselves” meet the promise to make learning mathematics a game like learning to play chess by software.

In order to get prototypes ready for field tests, understanding by stakeholders in STEM education is needed, private demos of Isabelle/ISAC are welcome.
Conclusions

TP technology provides mechanical explanations due to
  • principal benefits
  • added value of implementation
  • and “next step guidance”

of various kinds on different language layers — all explanations come on users’ request!

Field tests will show, whether “systems that explain themselves” meet the promise to make learning mathematics a game like learning to play chess by software.

In order to get prototypes ready for field tests, understanding by stakeholders in STEM education is needed, private demos of Isabelle/ISAC are welcome.
TP technology provides mechanical explanations due to

- principal benefits
- added value of implementation
- and “next step guidance”

of various kinds on different language layers — all explanations come on users’ request!

Field tests will show, whether “systems that explain themselves” meet the promise to make learning mathematics a game like learning to play chess by software.

In order to get prototypes ready for field tests, understanding by stakeholders in STEM education is needed, private demos of Isabelle/ISAC are welcome.
Thank you for Attention!