

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 TODO, 2018

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

Conclusions

- 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”?

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

Conclusions

- 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), Mathtoy, *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”?

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

Conclusions

- 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), Mathtoy, *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

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

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), Mathtoy, *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”?

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

Conclusions

- 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”?

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

Conclusions

- 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), Mathtoy, *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”?

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

Conclusions

- 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”?

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

Conclusions

- 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), Mathtoy, *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”?

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

Conclusions

- 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), Mathtoy, *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

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

Conclusions

1 “Systems that Explain Themselves”?

2 Mechanical Explanation and Language Layers

Term Language

Proof Language

Specification Language

“Next step guidance”

Programming Language

3 Conclusions

Term Language

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

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

Term Language

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

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

Term Language

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

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

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

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:

The screenshot displays the ISAC front-end interface with several panels:

- Example browser:** Shows "Abb.7.59" with a diagram of a cantilever beam of length L under a uniformly distributed load q_0 . The text asks to determine the deflection line for a cantilever beam of length L with a constant distributed load q_0 using boundary conditions $Q(0) = q_0 \cdot L$, $M_b(L) = 0$, $y(0) = 0$, and $y'(0) = 0$.
- Worksheet:** Contains a sequence of steps:
 - Problem (Biegelinie, [Biegelinien])
 - Problem (Biegelinie, [vonBelastungZu, Biegelinien])
 - $qq\ x = q\ 0$
 - $-qq\ x = -q\ 0$ (Rewrite("Belastung Querkraft",""))
 - $Q'x = -q\ 0$
 - Integrate (-q 0, x)
 - $Qx = \text{Integral} -q\ 0\ D\ x$
 - $Qx = \text{Integral} -1 * q\ 0\ D\ x$
 - $Qx = c + -1 * q\ 0 * x$
- Theory browser:** Displays the differential equation $V - g\ dx - (V + dv) = 0$ and the boundary condition $\frac{dV}{dx} = -g$. It explains that the negative load function is the derivative of the shear force, and the slope of the shear force is influenced by the load height. It also shows the equilibrium condition $\Sigma M = 0$ and the differential equation $M + V \cdot dx - g \cdot dx \cdot \frac{dx}{2} - (M + dM) = 0$.

Figure: Conventional worksheet on ISAC's front-end

Proof Language

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

- 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

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

- 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

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

1 “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

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

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

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

- 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

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

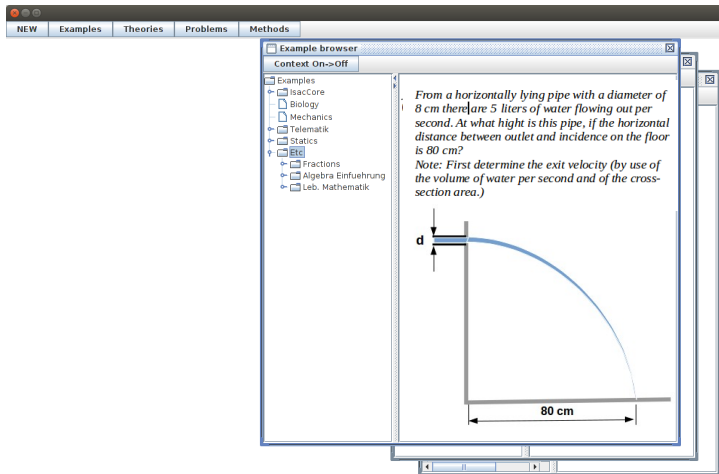
Step Guidance

Prog. Language

Conclusions

- 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



Start Example

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

NEW Examples Theories Problems Methods NEXT AUTO

Worksheet

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.)

d

80 cm

Problem modelled ok

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

The screenshot shows a software interface with a menu bar (NEW, Examples, Theories, Problems, Methods) and buttons for NEXT and AUTO. A 'Worksheet' window is open, displaying the following text:

Model:
Given : Diameter $d = 8\text{ cm}$, FlowRate $\phi = 5\text{ l/s}$,
HorizontalDistance $s = 80\text{ cm}$
Where : $d > 0 \wedge \phi > 0 \wedge s > 0$
Find : HeightOfPipe h

To the right of the text is a diagram of a horizontally lying pipe with diameter d . Water is flowing out of the pipe, forming a parabolic arc that lands on a horizontal surface at a distance of 80 cm from the pipe's outlet. The diagram illustrates the trajectory of the water jet.

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

Explain itself?

Lang. Layers

Term Language

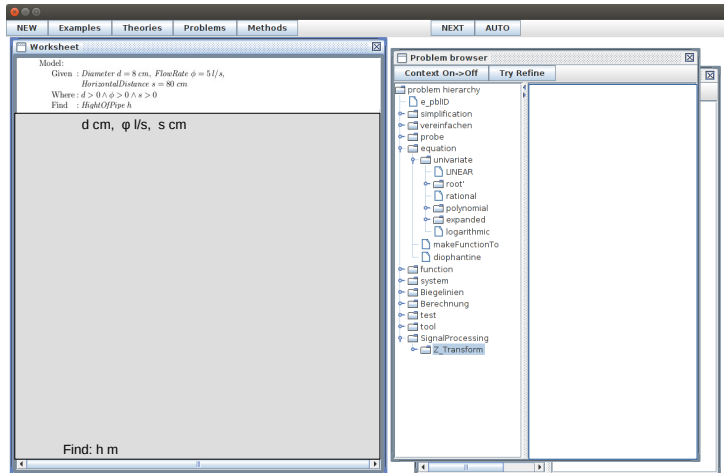
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Select sub-problems

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

The screenshot shows a software interface with two main windows. The left window, titled 'Worksheet', contains a problem statement and a box labeled 'Problem [rational, equation]'. The right window, titled 'Problem browser', shows a hierarchy of problem types with 'equation' selected. An orange arrow points from the 'equation' node in the hierarchy to the 'Problem [rational, equation]' box in the worksheet.

Worksheet Content:

Model:
 Given : Diameter $d = 8 \text{ cm}$, FlowRate $\phi = 5 \text{ l/s}$,
 HorizontalDistance $s = 80 \text{ cm}$
 Where : $d > 0 \wedge \phi > 0 \wedge s > 0$
 Find : HeightOfPipe h

$d \text{ cm}, \phi \text{ l/s}, s \text{ cm}$

Problem [rational, equation]

Find: $h \text{ m}$

Problem browser Content:

Context On->Off Try Refine

problem hierarchy

- e_pblid
 - simplification
 - vereinfachen
 - probe
 - equation
 - univariate
 - LINEAR
 - root
 - rational
 - polynomial
 - expanded
 - logarithmic
 - makeFunctionTo
 - diophantine
- function
- system
- Biegelinien
- Berechnung
- test
- tool
- SignalProcessing
 - Z_Transform

solve (e_e, v_v)

Model:

Given:	equality e_e solveFor v_v
Where:	e_e is_rateequation_in v_v
Find:	solutions v_v'i
Relate:	

Select sub-problems

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

The screenshot displays a software interface with two main windows. The top window, titled "Worksheet", contains a problem statement in German. It specifies a diameter $d = 8 \text{ cm}$, a flow rate $\phi = 5 \text{ l/s}$, and a horizontal distance $s = 80 \text{ cm}$. It asks to find the height h of a pipe. Below the text, there are two boxes: "Problem [rational, equation]" and "Problem [velocity-space-time, find-time]". The second box contains the equation $v = \frac{s}{t}$. The bottom window, titled "Problem browser", shows a hierarchical tree of problem types. The tree includes categories like "problem hierarchy", "e_pblid", "simplification", "vereinfachen", "probe", "equation", "univariate", "LINEAR", "root", "rational", "polynomial", "expanded", "logarithmic", "makeFunctionTo", "diophantine", "function", "system", "Biegelinien", "Berechnung", "test", "tool", "SignalProcessing", and "Z_Transform". Two red arrows point from the "Problem browser" window to the two problem boxes in the "Worksheet" window, indicating the selection of sub-problems.

NEW Examples Theories Problems Methods NEXT AUTO

Worksheet

Model:
Given : Diameter $d = 8 \text{ cm}$, FlowRate $\phi = 5 \text{ l/s}$,
HorizontalDistance $s = 80 \text{ cm}$
Where : $d > 0 \wedge \phi > 0 \wedge s > 0$
Find : HeightOfPipe h

$d \text{ cm}, \phi \text{ l/s}, s \text{ cm}$

Problem [rational, equation]

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

$v = \frac{s}{t}$

Find: $h \text{ m}$

Problem browser

Context On->Off Try Refine

- problem hierarchy
 - e_pblid
 - simplification
 - vereinfachen
 - probe
 - equation
 - univariate
 - LINEAR
 - root
 - rational
 - polynomial
 - expanded
 - logarithmic
 - makeFunctionTo
 - diophantine
 - function
 - system
 - Biegelinien
 - Berechnung
 - test
 - tool
 - SignalProcessing
 - Z_Transform

Select sub-problems

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

The screenshot shows a software interface with two main windows. The top window is titled "Worksheet" and contains a model description:

Model:
 Given : Diameter $d = 8$ cm, FlowRate $\phi = 5$ l/s,
 HorizontalDistance $s = 80$ cm
 Where : $d > 0 \wedge \phi > 0 \wedge s > 0$
 Find : HeightOfPipe h

Below the model, the variables d cm, ϕ l/s, s cm are listed. Three sub-problems are identified and linked by red arrows to the "Problem browser" window:

- Problem [rational, equation] (linked to the "equation" node in the browser)
- Problem [velocity-space-time, find-time] (linked to the "diophantine" node in the browser)
- Problem [flow-rate, find-velocity] (linked to the "function" node in the browser)

The "Problem browser" window shows a hierarchy of problem types:

- problem hierarchy
 - e_pblid
 - simplification
 - vereinfachen
 - probe
 - equation
 - univariate
 - LINEAR
 - root'
 - rational
 - polynomial
 - expanded
 - logarithmic
 - makeFunctionTo
 - diophantine
 - function
 - system
 - Biegelinien
 - Berechnung
 - test
 - tool
 - SignalProcessing
 - Z_Transform

The bottom window is titled "Find: h m".

Select sub-problems

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

The screenshot shows a software interface with two main windows. The left window, titled "Worksheet", contains a problem statement and its sub-problems. The right window, titled "Problem browser", shows a hierarchy of problems.

Worksheet Window:

Model:
 Given : Diameter $d = 8 \text{ cm}$, FlowRate $\phi = 5 \text{ l/s}$,
 HorizontalDistance $s = 80 \text{ cm}$
 Where : $d > 0 \wedge \phi > 0 \wedge s > 0$
 Find : $HeightOfPipe \ h$

$d \text{ cm}, \phi \text{ l/s}, s \text{ cm}$

Problem [rational, equation]

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

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

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

Find: $h \text{ m}$

Problem browser Window:

Context On->Off Try Refine

problem hierarchy

- e_pblid
- simplification
- vereinfachen
- probe
- equation
 - univariate
 - LINEAR
 - root'
 - rational
 - polynomial
 - expanded
 - logarithmic
 - makeFunctionTo
 - diophantine
- function
- system
- Biegelinien
- Berechnung
- test
- tool
- SignalProcessing
 - Z_Transform

Four red arrows point from the "Problem browser" window to the sub-problem boxes in the "Worksheet" window, indicating the selection of sub-problems.

Delete irrelevant sub-problems

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

The screenshot shows a software interface with two main windows. The top window is titled "Worksheet" and contains a problem statement in German. The bottom window is titled "Problem browser" and shows a tree view of problem categories.

Worksheet Window:

Model:
 Given : Diameter $d = 8 \text{ cm}$, FlowRate $\phi = 5 \text{ l/s}$,
 HorizontalDistance $s = 80 \text{ cm}$
 Where : $d > 0 \wedge \phi > 0 \wedge s > 0$
 Find : HeightOfPipe h

$d \text{ cm}, \phi \text{ l/s}, s \text{ cm}$

~~Problem [rational equation]~~

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: $h \text{ m}$

Problem browser Window:

Context On->Off Try Refine

- problem hierarchy
 - e_pblid
 - simplification
 - vereinfachen
 - probe
 - equation
 - univariate
 - LINEAR
 - root
 - rational
 - polynomial
 - expanded
 - logarithmic
 - makeFunctionTo
 - diophantine
 - function
 - system
 - Biegelinien
 - Berechnung
 - test
 - tool
 - SignalProcessing
 - Z_Transform

Select relevant sub-problems

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

The screenshot displays a software interface with two main windows. The top window, titled "Worksheet", contains a model description: "Given : Diameter $d = 8$ cm, FlowRate $\phi = 5$ l/s, HorizontalDistance $s = 80$ cm. Where : $d > 0 \wedge \phi > 0 \wedge s > 0$. Find : HeightOfPipe h ". Below this, the units "d cm, ϕ l/s, s cm" are listed. Four problem boxes are shown: "Problem [area-of-circle]" with the formula $A_{circle} = \left(\frac{d}{2}\right)^2 \cdot \pi$, "Problem [velocity-space-time]" with $v = \frac{s}{t}$, "Problem [flow-rate, find-velocity]" with $v = \frac{\phi}{A_{circle}}$, and "Problem [free-fall]" with $h = \frac{g}{2} \cdot t^2$. The bottom of the worksheet says "Find: h m". The right window, titled "Problem browser", shows a tree structure of problem categories. An orange arrow points from the "equation" category in the browser to the "Problem [area-of-circle]" box in the worksheet.

Worksheet

Model:

Given : Diameter $d = 8$ cm, FlowRate $\phi = 5$ l/s,
HorizontalDistance $s = 80$ cm
Where : $d > 0 \wedge \phi > 0 \wedge s > 0$
Find : HeightOfPipe h

d cm, ϕ 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]
 $v = \frac{\phi}{A_{circle}}$

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

Find: h m

Problem browser

Context On->Off Try Refine

- problem hierarchy
 - e_pblid
 - simplification
 - vereinfachen
 - probe
 - equation
 - univariate
 - LINEAR
 - root
 - rational
 - polynomial
 - expanded
 - logarithmic
 - makeFunctionTo
 - diophantine
 - function
 - system
 - Biegelinien
 - Berechnung
 - test
 - tool
 - SignalProcessing
 - Z_Transform

What is given / has to be found?

Explain itself?

Lang. Layers

Term Language

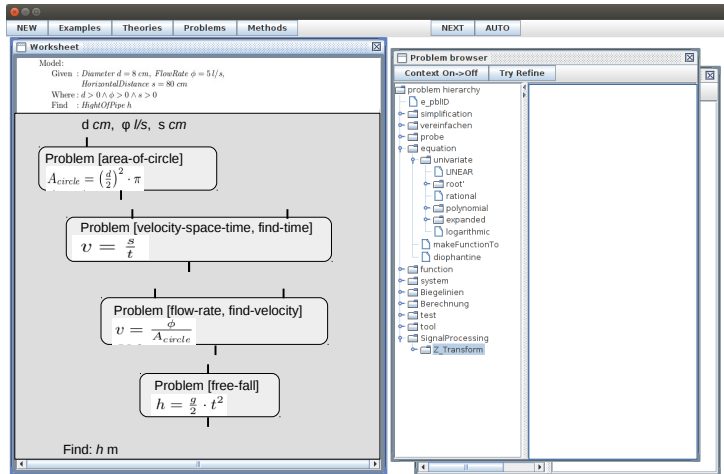
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



What is given / has to be found?

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

The screenshot displays a software interface for mechanised justification, divided into two main panes.

Worksheet Pane:

- Model:**
 - Given : Diameter $d = 8 \text{ cm}$, FlowRate $\phi = 5 \text{ l/s}$, HorizontalDistance $s = 80 \text{ cm}$
 - Where : $d > 0 \wedge \phi > 0 \wedge s > 0$
 - Find : HeightOfPipe h
- Problem Hierarchy:**
 - Initial variables: $d \text{ cm}, \phi \text{ l/s}, s \text{ cm}$
 - Problem [area-of-circle]**: $A_{\text{circle}} = \left(\frac{d}{2}\right)^2 \cdot \pi$. Variables: $A \text{ cm}^2$.
 - Problem [velocity-space-time, find-time]**: $v = \frac{s}{t}$. Variables: $v \text{ m/s}, s \text{ m}, t \text{ s}$.
 - Problem [flow-rate, find-velocity]**: $v = \frac{\phi}{A_{\text{circle}}}$. Variables: $A \text{ m}^2, \phi \text{ m}^3/\text{s}, v \text{ m/s}$.
 - Problem [free-fall]**: $h = \frac{g}{2} \cdot t^2$. Variables: $t \text{ s}, h \text{ m}$.
 - Find:** $h \text{ m}$

Problem browser Pane:

- Context On->Off** / **Try Refine**
- problem hierarchy**
 - e_pblid
 - simplification
 - vereinfachen
 - probe
 - equation
 - univariate
 - LINEAR
 - root
 - rational
 - polynomial
 - expanded
 - logarithmic
 - makeFunctionTo
 - diophantine
 - function
 - system
 - Biegelinien
 - Berechnung
 - test
 - tool
 - SignalProcessing
 - Z_Transform

Connect “Given” and “Find”

Explain itself?

Lang. Layers

Term Language

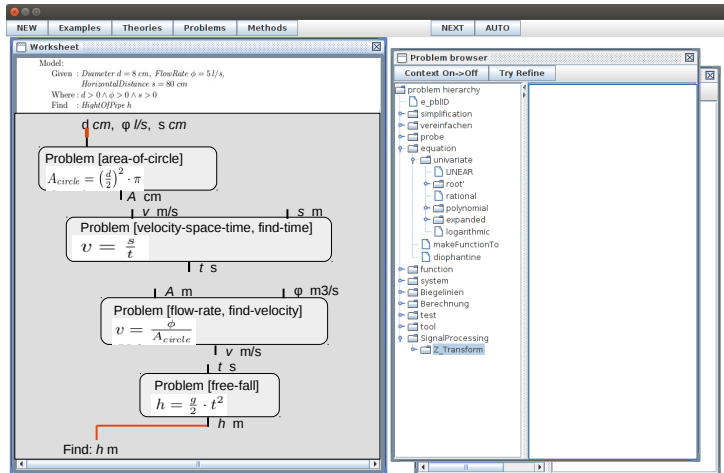
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Connect “Given” and “Find”

Explain itself?

Lang. Layers

Term Language

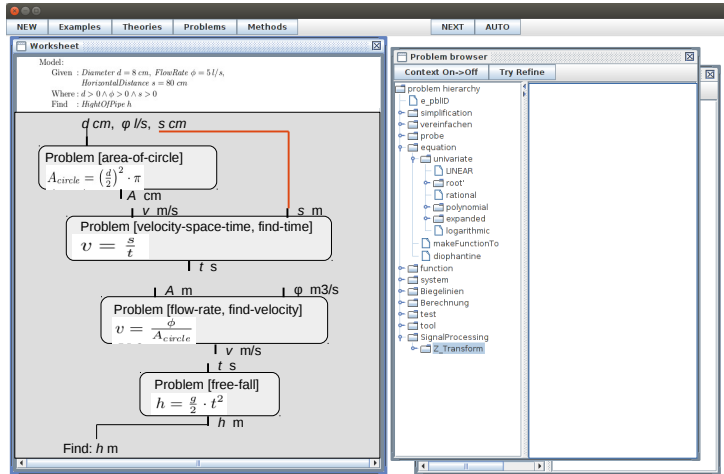
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Connect “Given” and “Find”

Explain itself?

Lang. Layers

Term Language

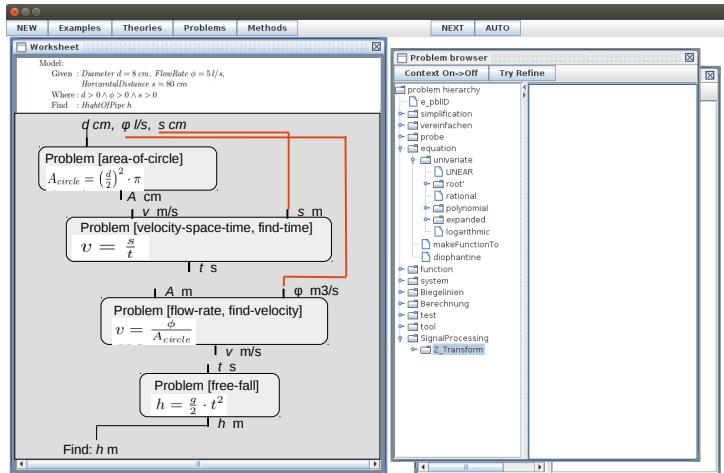
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Connect “Given” and “Find”

Explain itself?

Lang. Layers

Term Language

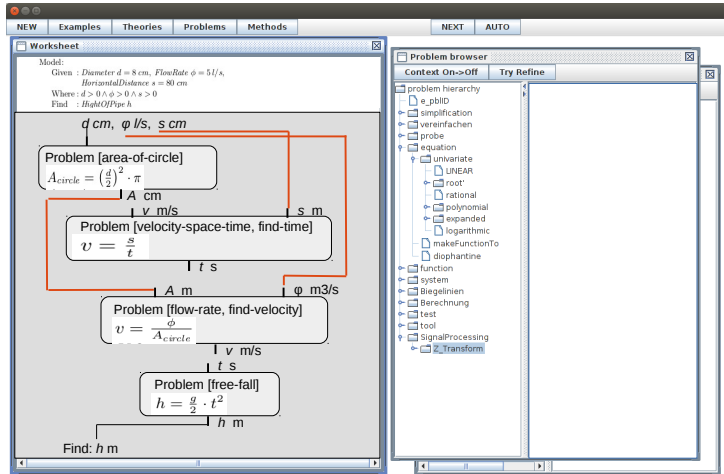
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Connect “Given” and “Find”

Explain itself?

Lang. Layers

Term Language

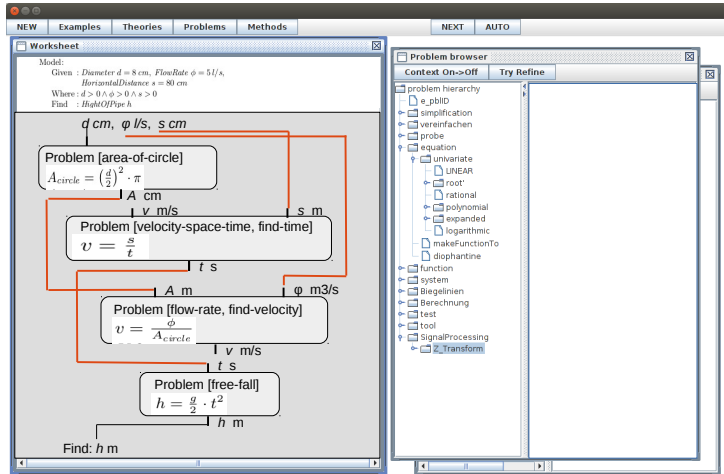
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Dangling connection ???

Explain itself?

Lang. Layers

Term Language

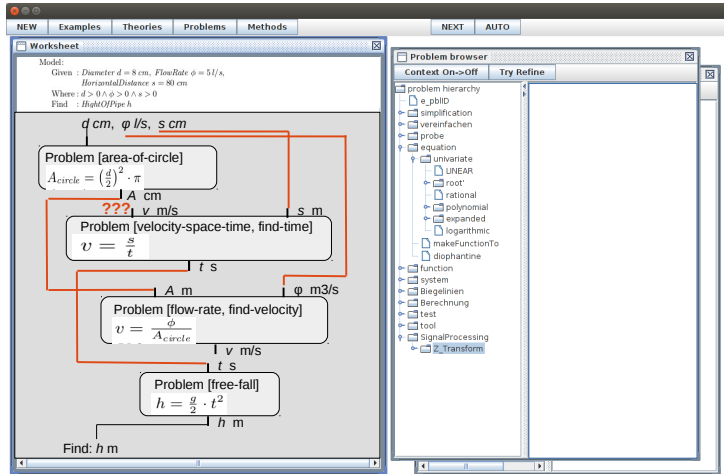
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Rearrange sub-problems

Explain itself?

Lang. Layers

Term Language

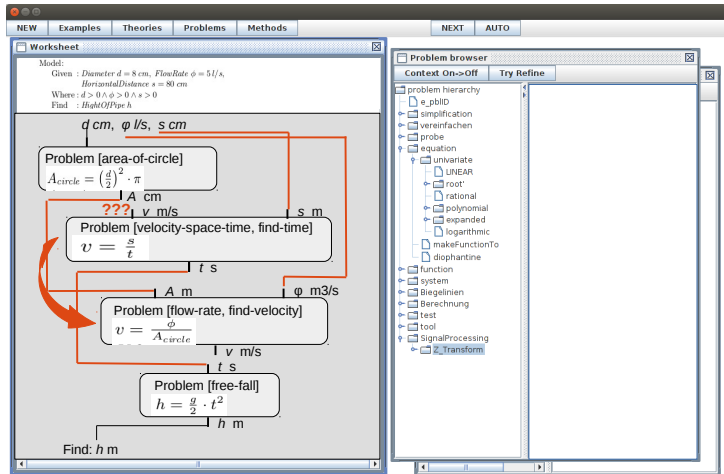
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Flipped two sub-problems

Explain itself?

Lang. Layers

Term Language

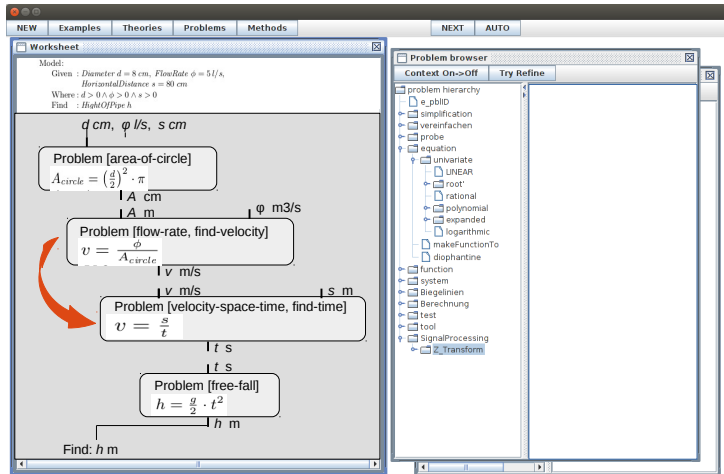
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Connect “Given” and “Find”

Explain itself?

Lang. Layers

Term Language

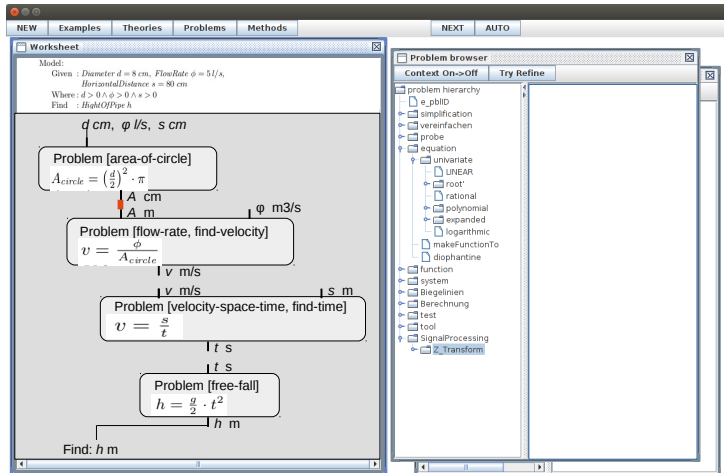
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Connect “Given” and “Find”

Explain itself?

Lang. Layers

Term Language

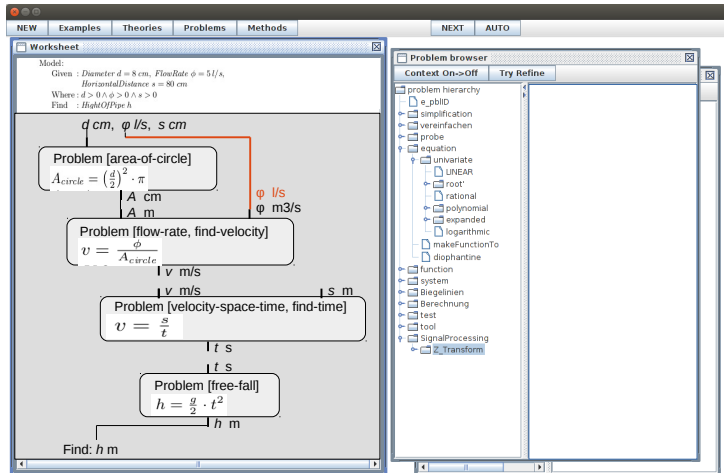
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Connect “Given” and “Find”

Explain itself?

Lang. Layers

Term Language

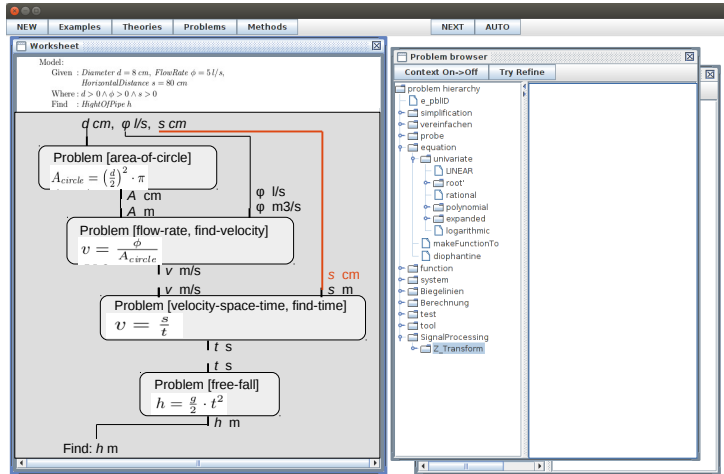
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Connect “Given” and “Find”

Explain itself?

Lang. Layers

Term Language

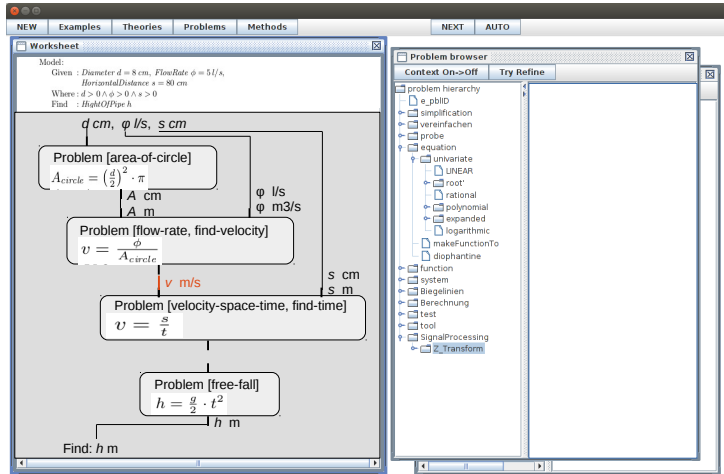
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Connect “Given” and “Find”

Explain itself?

Lang. Layers

Term Language

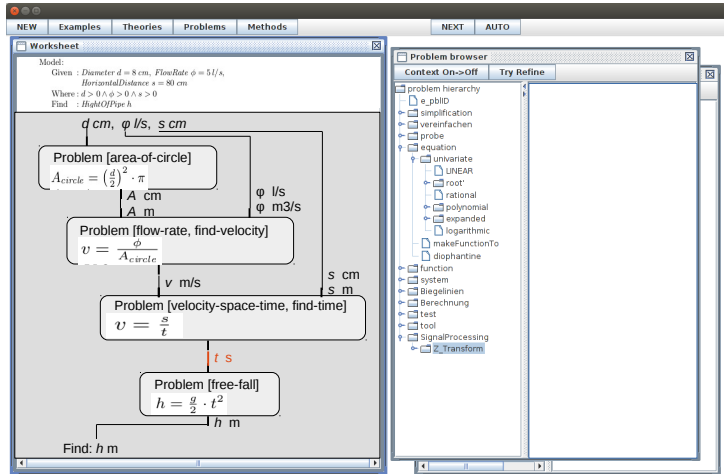
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



All connections finished

Explain itself?

Lang. Layers

Term Language

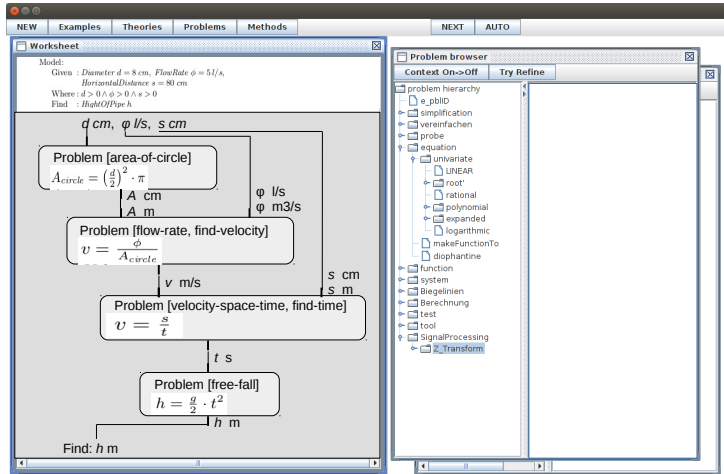
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Care about unit conversions

Explain itself?

Lang. Layers

Term Language

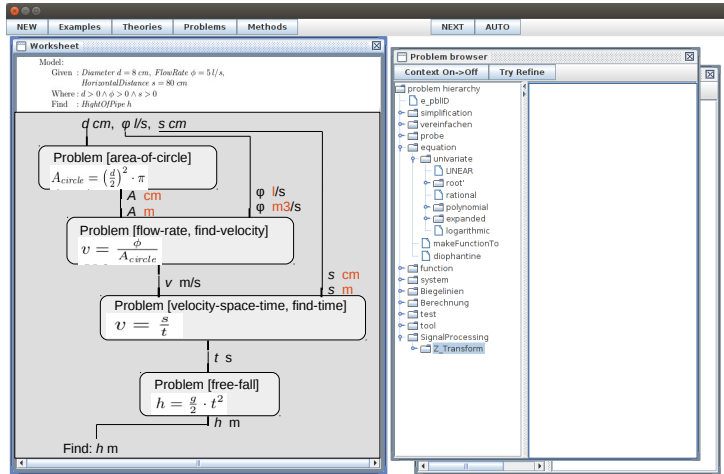
Proof Language

Specification

Step Guidance

Prog. Language

Conclusions



Solution with units only

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

The screenshot displays the Isabelle/HOL proof assistant interface. The main window, titled 'Worksheet', contains a mechanical solution for a problem involving fluid flow in a pipe. The solution is structured as follows:

Model:
 Given : Diameter $d = 8 \text{ cm}$, FlowRate $\phi = 5 \text{ l/s}$,
 HorizontalDistance $s = 80 \text{ cm}$
 Where : $d > 0 \wedge \phi > 0 \wedge s > 0$
 Find : HeightOfPipe h

Solution:

Problem [area-of-circle]:
 $A_{\text{circle}} \text{ cm}$ Unit_conversion cm2_m2
 $A_{\text{circle}} \text{ m}$
 $\phi = 5 \frac{\text{l}}{\text{s}}$ Unit_conversion l_m3
 $\phi = 0,005 \frac{\text{m}^3}{\text{s}}$
 Problem [flow-rate, find-velocity]:
 $v \frac{\text{m}}{\text{s}}$
 $s = 80 \text{ cm}$ Unit_conversion cm_m
 $s = 0,8 \text{ m}$
 Problem [velocity-space-time, find-time]:
 $t \frac{\text{m}}{\text{s}}$
 Problem [free-fall]:
 $h \text{ m}$

The right-hand pane shows the 'Problem browser' with a tree view of the problem hierarchy. The selected item is 'Z_Transform' under the 'SignalProcessing' node.

Solution complete

The screenshot displays two windows from a software application. The 'Worksheet' window on the left contains a problem statement and its solution. The problem involves a pipe with a diameter of 8 cm and a flow rate of 5 l/s, with a horizontal distance of 80 cm. The solution includes calculations for the area of the circle, unit conversions, and the final height of the pipe.

The 'Problem browser' window on the right shows a tree structure of problem-solving methods. The tree is organized into categories such as 'problem hierarchy', 'univariate', 'function', 'system', 'Biegelinien', 'Berechnung', 'test', 'tool', and 'SignalProcessing'. The 'Z_Transform' method is highlighted under the 'SignalProcessing' category.

Worksheet

Model:
 Given : Diameter $d = 8 \text{ cm}$, FlowRate $\phi = 5 \text{ l/s}$,
 HorizontalDistance $s = 80 \text{ cm}$
 Where : $d > 0 \wedge \phi > 0 \wedge s > 0$
 Find : HeightOfPipe h

Solution:

Problem [area-of-circle]
 $A_{\text{circle}} = 50 \text{ cm}^2$ Unit_conversion $\text{cm}^2_m^2$
 $A_{\text{circle}} = 0,005 \text{ m}^2$ Take.given ϕ
 $\phi = 5 \frac{\text{l}}{\text{s}}$ Unit_conversion l_m^3
 $\phi = 0,005 \frac{\text{m}^3}{\text{s}}$
 Problem [flow-rate, find-velocity]
 $v = 1 \frac{\text{m}}{\text{s}}$ Unit_conversion cm_m
 $s = 80 \text{ cm}$
 $s = 0,8 \text{ m}$
 Problem [velocity-space-time, find-time]
 $t = 0,8 \frac{\text{m}}{\text{s}}$
 Problem [free-fall]:
 $h = 3,2 \text{ m}$ Check_postcond [composed, movement, no-6]

Problem browser

- problem hierarchy
 - e_pblid
 - simplification
 - vereinfachen
 - probe
 - equation
 - univariate
 - LINEAR
 - root'
 - rational
 - polynomial
 - expanded
 - logarithmic
 - makeFunctionTo
 - diophantine
 - function
 - system
 - Biegelinien
 - Berechnung
 - test
 - tool
 - SignalProcessing
 - Z_Transform

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

Conclusions

1 “Systems that Explain Themselves”?

2 Mechanical Explanation and Language Layers

Term Language

Proof Language

Specification Language

“Next step guidance”

Programming Language

3 Conclusions

“Next step guidance” ...

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

- ... 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” ...

Explain itself?

Lang. Layers

Term Language

Proof Language

Specification

Step Guidance

Prog. Language

Conclusions

- ... 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.

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

Conclusions

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 ...

Explain itself?

Lang. Layers

Term Language
Proof Language
Specification
Step Guidance
Prog. Language

Conclusions

... 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.

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.

Thank you for Attention!