On Conceptual Design of Educational Mathematics Assistants

Software as a Model of Mathematics

Walther Neuper

Institute for Softwaretechnology
Graz University of Technology

ÖMG-Kongress – Jahrestagung der DMV
TU Graz Sept.09
Outline

1. Software as a model of mathematics
2. Human readable implementation of math
   - Demo: readable knowledge
   - Demo: readable proofs
   - Demo: “transparent” systems w.r.t. knowledge
   - Demo: “interactive” systems w.r.t. steps in calculations
Software models math

1. Elements of the *language of mathematics* receive their meaning from operations admissible on them.

2. *Software can implement all* formal elements of math, elements concerning knowledge and algorithms.

3. If such software were implemented human readably, one could learn just by interacting with this software.
Software models math

1. Elements of the *language of mathematics* receive their meaning from operations admissible on them.

2. *Software can implement all* formal elements of math, elements concerning knowledge and algorithms.

3. If such software were implemented human readably, one could learn just by interacting with this software.
Software models math

1. Elements of the *language of mathematics* receive their meaning from operations admissible on them.

2. *Software can implement all* formal elements of math, elements concerning knowledge and algorithms.

3. If such software were implemented human readably, one could learn just by interacting with this software.
Software models math

1. Elements of the *language of mathematics* receive their meaning from operations admissible on them.

2. *Software can implement all* formal elements of math, elements concerning knowledge and algorithms.

3. If such software were implemented *human readably*, one could learn just by interacting with this software.
Outline

1. Software as a model of mathematics

2. Human readable implementation of math
   - Demo: readable knowledge
   - Demo: readable proofs
   - Demo: “transparent” systems w.r.t. knowledge
   - Demo: “interactive” systems w.r.t. steps in calculations
**Demo: readable knowledge**

http://isabelle.in.tum.de/dist/library/HOL/Int.html

begin

subsection {* The equivalence relation underlying the integers *}

definition intrel :: "((nat × nat) × (nat × nat)) set" where
"intrel = ((x, y), (u, v)) | x y u v. x + v = u + y"

...

subsection {*Construction of the Integers*}

lemma intrel_iff [simp]: "(((x,y),(u,v)) ∈ intrel) = (x+v = u+y)"
by (simp add: intrel_def)

...

... can be read by humans and interpreted by software!
Demo: readable knowledge

http://isabelle.in.tum.de/dist/library/HOL/Int.html

begin

subsection {* The equivalence relation underlying the integers *}

definition intrel :: "((nat × nat) × (nat × nat)) set" where "intrel = ((x, y), (u, v)) | x y u v. x + v = u + y"

subsection {*Construction of the Integers*}

lemma intrel_iff [simp]: "(((x,y),(u,v)) ∈ intrel) = (x+v = u+y)" by (simp add: intrel_def)

... can be read by humans and interpreted by software!
Outline

1. Software as a model of mathematics

2. Human readable implementation of math
   - Demo: readable knowledge
   - Demo: readable proofs
   - Demo: “transparent” systems w.r.t. knowledge
   - Demo: “interactive” systems w.r.t. steps in calculations
Demo: readable knowledge

http://isabelle.in.tum.de/dist/library/HOL/Library/Primes.html

lemma even_square: assumes e: "even (n::nat)"
shows "∃x. n ^ 2 = 4*x"
proof-
  from e have "2 dvd n" by presburger
  then obtain k where k: "n = 2*k" using dvd_def by auto
  hence "n^2 = 4* (k^2)" by (simp add: power2_eq_square)
  thus ?thesis by blast
qed

...can be read by humans and interpreted by software!
Demo: readable knowledge

http://isabelle.in.tum.de/dist/library/HOL/Library/Primes.html

lemma even_square: assumes e: "even (n::nat)"
shows "∃x. n^2 = 4*x"
proof-
  from e have "2 dvd n" by presburger
  then obtain k where k: "n = 2*k" using dvd_def by auto
  hence "n^2 = 4*(k^2)" by (simp add: power2_eq_square)
  thus ?thesis by blast
qed

...can be read by humans and interpreted by software!
Outline

1. Software as a model of mathematics

2. Human readable implementation of math
   - Demo: readable knowledge
   - Demo: readable proofs
   - Demo: “transparent” systems w.r.t. knowledge
   - Demo: “interactive” systems w.r.t. steps in calculations
"Transparent" Systems

- Software can make transparent (virtually all) knowledge underlying a certain calculation.
- Investigation of the knowledge is up to a free decision by the user.
- Investigation is exhaustive in the formal scope and comprises deductive dependencies in all details.
Outline

1. Software as a model of mathematics

2. Human readable implementation of math
   - Demo: readable knowledge
   - Demo: readable proofs
   - Demo: “transparent” systems w.r.t. knowledge
   - Demo: “interactive” systems w.r.t. steps in calculations
“Single-stepping” Systems

- The system “knows” the next step.
- The learner can input a rule, and the system checks if applicable or not.
- The learner can input a formula, and the system tries to find a derivation.
"Transparent single-stepping systems" for step-wise problem solving in applied mathematics shall

- be an ubiquitous service wherever formulas occur
  - embedded in wikis, elearning content, virtual laboratories
- support individual learning within one course
  - e.g. just watching the system . . . . . . indepth investigation
- establish a continuum from high-school to university
  - from intro of variables to advanced courses in engineering
- provide active experience about mechanization of thinking
  - as the central thinking technology in science . . . as a game