Technology of Deduction for "Systems that Explain Themselves"

A new generation of educational software for engineering mathematics

Walther Neuper

IICM Institute for Information Systems and Computer Media
Graz University of Technology,
RISC und IIS at Johannes Kepler University Linz

ThEdu'17 at CADE 26
6. Aug. 2017

Outline

1. Misunderstandings ???
2. TP's benefits for educational math software
   - TP's knowledge is self-contained
   - each step is justified
   - . . . + computation: Lucas-Interpretation
   - TPs are open source
3. Promises of future software based on TP
   - Foster abstraction by operation
   - Cover all phases of problem solving
   - Support independent learning
   - Connect application — theory
4. Watch the demo at xx:xx !!!

Misunderstandings ???

- Proof assistants (TPs) are self-explanatory NO !

  Rather: use TP technology to *build new* SW !
- Computers can only compute (finite objects) NO !

  Rather: TPs proof Kepler Conjecture, verify SW, etc
- Human thought can*not* be mechanised YES/NO !

  Rather: *every*thing can be mechanised
  as soon as it is mathematised !
- "Systems that Explain Themselves" replace teachers NO !

  Rather: free teachers for non-mechanical aspects !

deductive technologies $\Longrightarrow$? $\Longleftarrow$ **learning mathematics**

Misunderstandings ???

- Proof assistants (TPs) are self-explanatory NO !

  Rather: use TP technology to *build new* SW !

- Computers can only compute (finite objects) NO !

  Rather: TPs proof Kepler Conjecture, verify SW, etc

- Human thought can*not* be mechanised YES/NO !

  Rather: *every*thing can be mechanised
  as soon as it is mathematised !

- "Systems that Explain Themselves" replace teachers NO !

  Rather: free teachers for non-mechanical aspects !

deductive technologies $\Longrightarrow$? $\Longleftarrow$ **learning mathematics**

Misunderstandings ???

- Proof assistants (TPs) are self-explanatory NO !

  Rather: use TP technology to *build new* SW !

- Computers can only compute (finite objects) NO !

  Rather: TPs proof Kepler Conjecture, verify SW, etc

- Human thought can*not* be mechanised YES/NO !

  Rather: *every*thing can be mechanised
  as soon as it is mathematised !

- "Systems that Explain Themselves" replace teachers NO !

  Rather: free teachers for non-mechanical aspects !

deductive technologies $\Longrightarrow$? $\Longleftarrow$ **learning mathematics**

Misunderstandings ???

- Proof assistants (TPs) are self-explanatory NO !

  Rather: use TP technology to *build new* SW !

- Computers can only compute (finite objects) NO !

  Rather: TPs proof Kepler Conjecture, verify SW, etc

- Human thought can*not* be mechanised YES/NO !

  Rather: *every*thing can be mechanised
  as soon as it is mathematised !

- "Systems that Explain Themselves" replace teachers NO !

  Rather: free teachers for non-mechanical aspects !

deductive technologies $\Longrightarrow$? $\Longleftarrow$ **learning mathematics**

Misunderstandings ???

- Proof assistants (TPs) are self-explanatory NO !

  Rather: use TP technology to *build new* SW !

- Computers can only compute (finite objects) NO !

  Rather: TPs proof Kepler Conjecture, verify SW, etc

- Human thought can*not* be mechanised YES/NO !

  Rather: *every*thing can be mechanised
  as soon as it is mathematised !

- "Systems that Explain Themselves" replace teachers NO !

  Rather: free teachers for non-mechanical aspects !

deductive technologies $\Longrightarrow$? $\Longleftarrow$ **learning mathematics**

Misunderstandings ???

- Proof assistants (TPs) are self-explanatory NO !

  Rather: use TP technology to *build new* SW !

- Computers can only compute (finite objects) NO !

  Rather: TPs proof Kepler Conjecture, verify SW, etc

- Human thought can*not* be mechanised YES/NO !

  Rather: *every*thing can be mechanised
  as soon as it is mathematised !

- "Systems that Explain Themselves" replace teachers NO !

  Rather: free teachers for non-mechanical aspects !

deductive technologies $\Longrightarrow$? $\Longleftarrow$ **learning mathematics**

Misunderstandings ???

- Proof assistants (TPs) are self-explanatory NO !

  Rather: use TP technology to *build new* SW !

- Computers can only compute (finite objects) NO !

  Rather: TPs proof Kepler Conjecture, verify SW, etc

- Human thought can*not* be mechanised YES/NO !

  Rather: *every*thing can be mechanised
  as soon as it is mathematised !

- "Systems that Explain Themselves" replace teachers NO !

  Rather: free teachers for non-mechanical aspects !

deductive technologies $\Longrightarrow$? $\Longleftarrow$ **learning mathematics**

Misunderstandings ???

- Proof assistants (TPs) are self-explanatory NO !

  Rather: use TP technology to *build new* SW !

- Computers can only compute (finite objects) NO !

  Rather: TPs proof Kepler Conjecture, verify SW, etc

- Human thought can*not* be mechanised YES/NO !

  Rather: *every*thing can be mechanised
  as soon as it is mathematised !

- "Systems that Explain Themselves" replace teachers NO !

  Rather: free teachers for non-mechanical aspects !

deductive technologies $\Longrightarrow$? $\Longleftarrow$ **learning mathematics**

Misunderstandings ???

- Proof assistants (TPs) are self-explanatory NO !

  Rather: use TP technology to *build new* SW !

- Computers can only compute (finite objects) NO !

  Rather: TPs proof Kepler Conjecture, verify SW, etc

- Human thought can*not* be mechanised YES/NO !

  Rather: *every*thing can be mechanised
  as soon as it is mathematised !

- "Systems that Explain Themselves" replace teachers NO !

  Rather: free teachers for non-mechanical aspects !

deductive technologies $\Longrightarrow$? $\Longleftarrow$ **learning mathematics**

Misunderstandings ???

- Proof assistants (TPs) are self-explanatory NO !

  Rather: use TP technology to *build new* SW !

- Computers can only compute (finite objects) NO !

  Rather: TPs proof Kepler Conjecture, verify SW, etc

- Human thought can*not* be mechanised YES/NO !

  Rather: *every*thing can be mechanised
  as soon as it is mathematised !

- "Systems that Explain Themselves" replace teachers NO !

  Rather: free teachers for non-mechanical aspects !

deductive technologies $\Longrightarrow$? $\Longleftarrow$ **learning mathematics**

Misunderstandings ???

- Proof assistants (TPs) are self-explanatory NO !

  Rather: use TP technology to *build new* SW !

- Computers can only compute (finite objects) NO !

  Rather: TPs proof Kepler Conjecture, verify SW, etc

- Human thought can*not* be mechanised YES/NO !

  Rather: *every*thing can be mechanised
  as soon as it is mathematised !

- "Systems that Explain Themselves" replace teachers NO !

  Rather: free teachers for non-mechanical aspects !

deductive technologies $\Longrightarrow$? $\Longleftarrow$ **learning mathematics**

Outline

1. Misunderstandings ???
2. TP's benefits for educational math software
   - TP's knowledge is self-contained
   - each step is justified
   - . . . + computation: Lucas-Interpretation
   - TPs are open source
3. Promises of future software based on TP
   - Foster abstraction by operation
   - Cover all phases of problem solving
   - Support independent learning
   - Connect application — theory
4. Watch the demo at xx:xx !!!

TP is self-contained

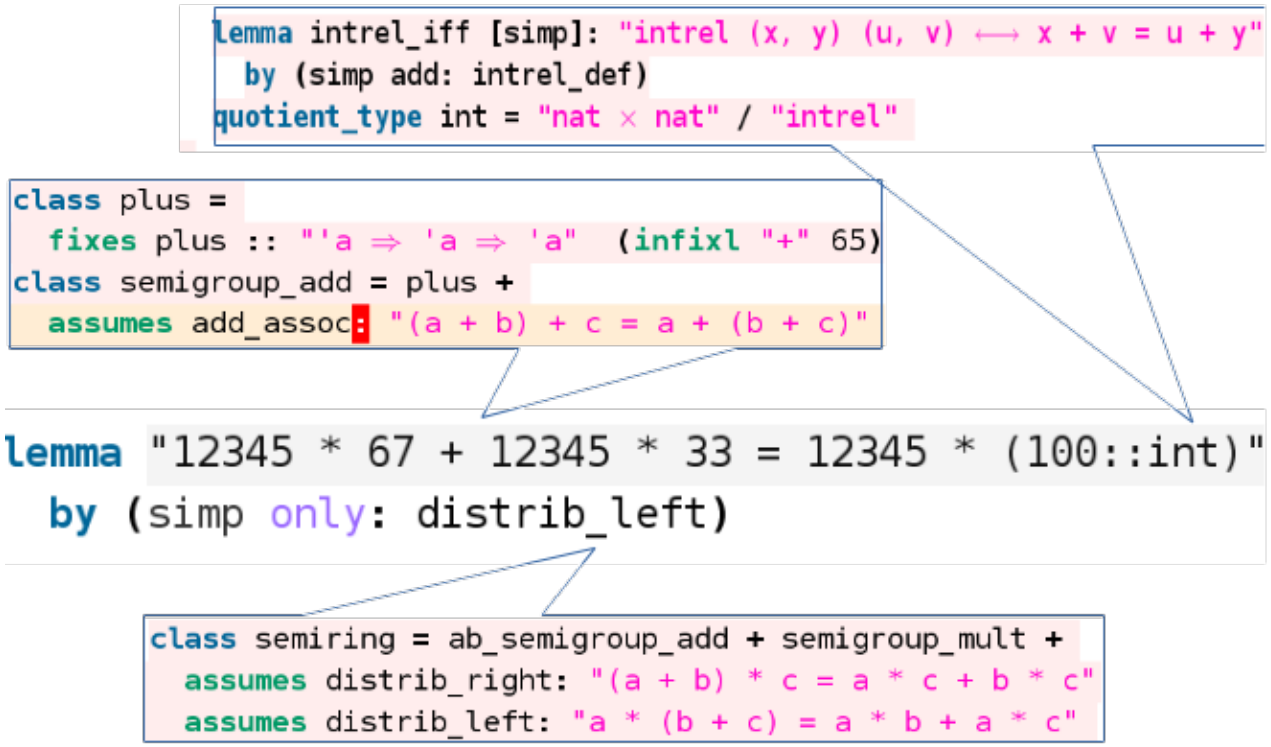


Figure : Knowledge underlying $12345 * 67 \ldots$ in TP Isabelle

Outline

1. Misunderstandings ???
2. TP's benefits for educational math software
   - TP's knowledge is self-contained
   - each step is justified
   - . . . + computation: Lucas-Interpretation
   - TPs are open source
3. Promises of future software based on TP
   - Foster abstraction by operation
   - Cover all phases of problem solving
   - Support independent learning
   - Connect application — theory
4. Watch the demo at xx:xx !!!

Watch the demo at xx:xx !!!

21 `Problem` [determine, 2-mass-oscillator, DiffEq]:
211 `Specification`:
212 `Solution`:

2121 forces of springs

2122 $[F_{c1} = c_1 x_1,\ F_{c2} = c_2(x_2 - x_1),\ F_{c3} = c_1 x_2]$

2123 forces of dampers

2124 $[F_{d1} = d\dot{x}_1,\ F_{d2} = d\dot{x}_2]$

2125 mass times acceleration equals sum of all forces

2126 $[m\ddot{x}_1 = -F_{c1} + F_{c2} - F_{d1},\ \ m\ddot{x}_2 = -F_{c2} - F_{c3} - F_{d2} + F_2]$

2127 `Substitute` $[F_{c1}, F_{c2}, F_{c3}, F_{d1}, F_{d2}]$

2128 $[m\ddot{x}_1 = -c_1 x_1 + c_2(c_2 - x_1) - d\dot{x}_1,\ \ m\ddot{x}_2 = -c_2(c_2 - x_1) - c_1 x_2 - d\dot{x}_2 + F_2]$

2129 `Rewrite_Set` normalise

212a $[m\ddot{x}_1 + d\dot{x}_1 + c_1 x_1 - c_2(x_2 - x_1) = 0,\ \ m\ddot{x}_2 + d\dot{x}_2 + c_2(x_2 - x_1) + c_1 x_1 = F_2]$

212b switch to vector representation

212c $$\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{pmatrix} + \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} + \begin{pmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_1 + c_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ F \end{pmatrix}$$

22 $$\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \ddot{x} + \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} \dot{x} + \begin{pmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_1 + c_2 \end{pmatrix} x = \begin{pmatrix} 0 \\ F \end{pmatrix}$$

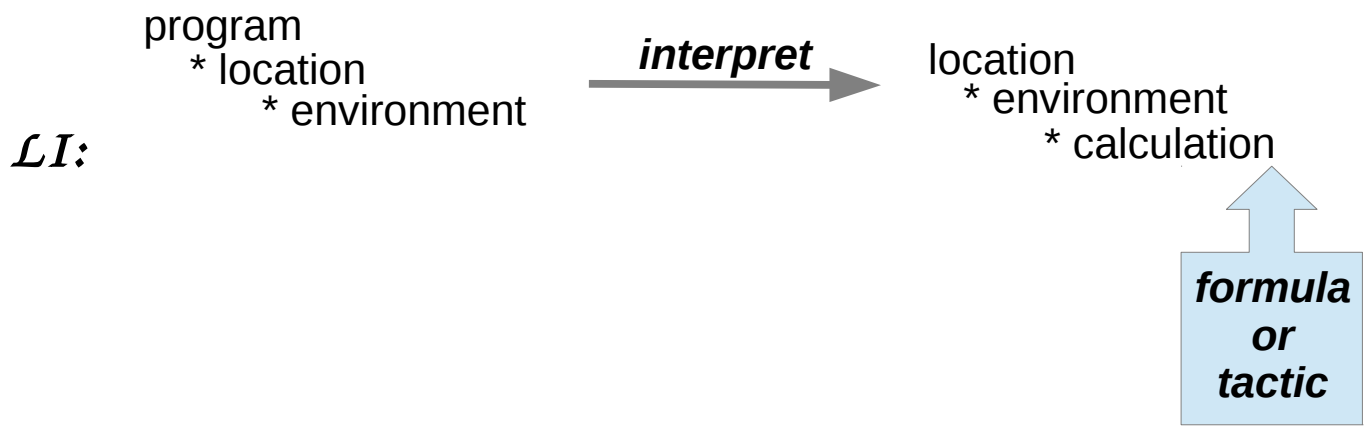
Outline

1. Misunderstandings ???
2. TP's benefits for educational math software
   - TP's knowledge is self-contained
   - each step is justified
   - . . . + computation: Lucas-Interpretation
   - TPs are open source
3. Promises of future software based on TP
   - Foster abstraction by operation
   - Cover all phases of problem solving
   - Support independent learning
   - Connect application — theory
4. Watch the demo at xx:xx !!!

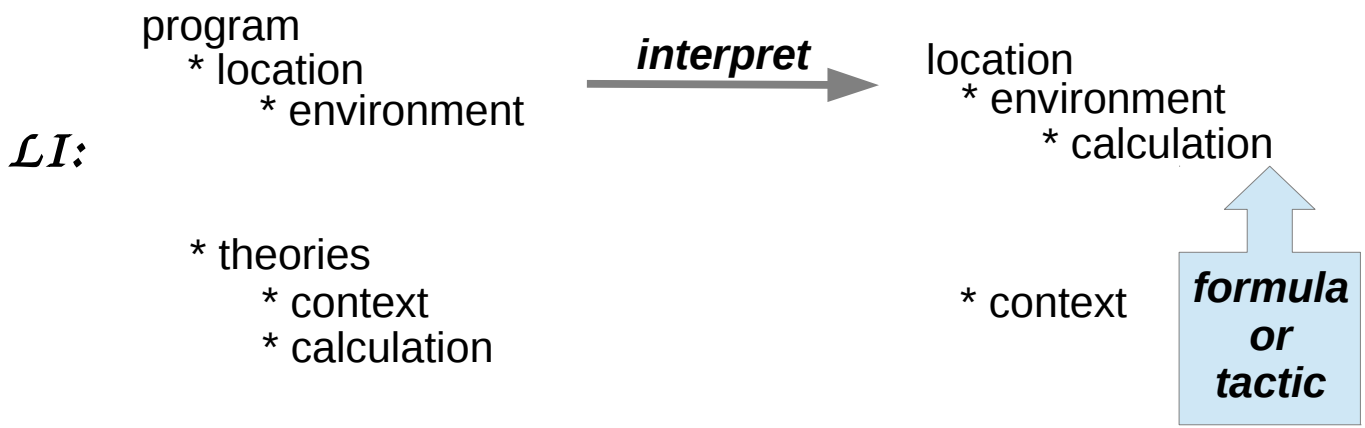
Lucas-Interpretation (LI)

Lucas-Interpretation (LI)

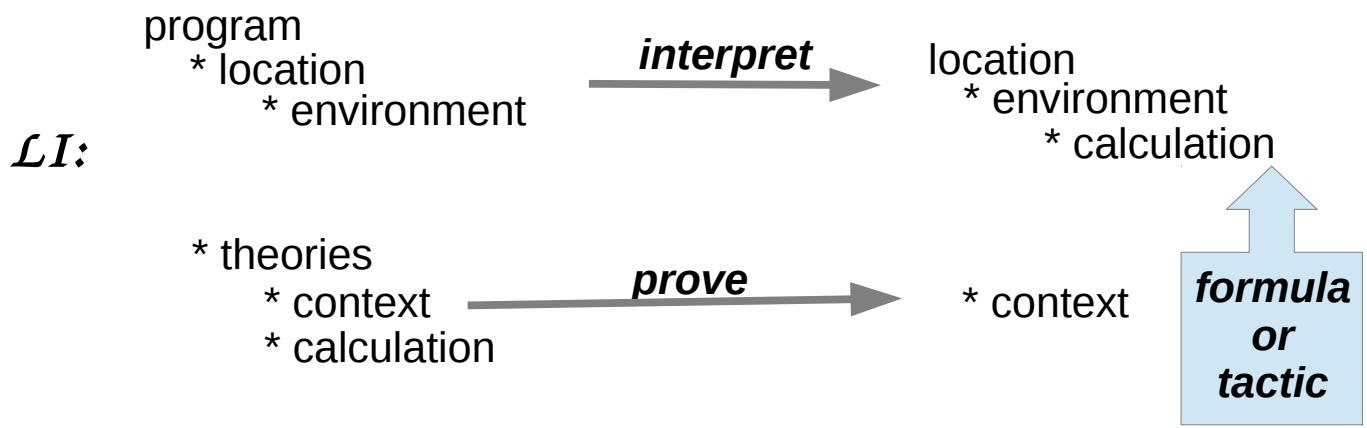
Lucas-Interpretation (LI)



Lucas-Interpretation (LI)

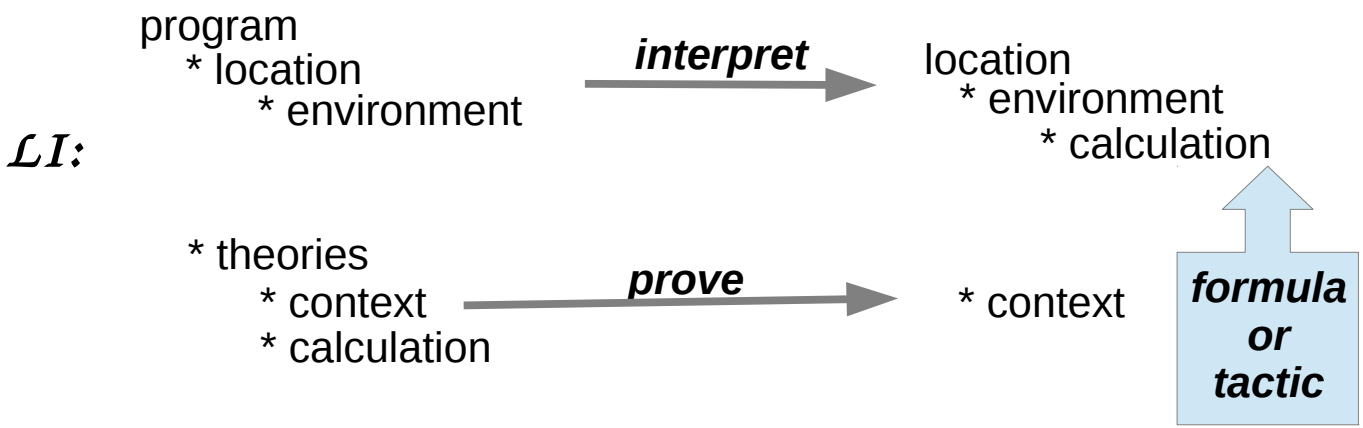


Lucas-Interpretation (LI)



Lucas-Interpretation (LI)

computation



deduction

Outline

1. Misunderstandings ???
2. TP's benefits for educational math software
   - TP's knowledge is self-contained
   - each step is justified
   - . . . + computation: Lucas-Interpretation
   - TPs are open source
3. Promises of future software based on TP
   - Foster abstraction by operation
   - Cover all phases of problem solving
   - Support independent learning
   - Connect application — theory
4. Watch the demo at xx:xx !!!

TPs are open source

Educational software must be

- free
- + open for teachers authoring own examples
- + open for teachers authoring own explanations
- + open for institutions authoring courses
- + open for institutions authoring educational strategies
- + open for further collaborative improvement

(e.g. *ISAC* shall remain an open source project !)

TPs are open source

Educational software must be

- free
- + open for teachers authoring own examples
- + open for teachers authoring own explanations
- + open for institutions authoring courses
- + open for institutions authoring educational strategies
- + open for further collaborative improvement

(e.g. $\mathcal{ISAC}$ shall remain an open source project !)

Outline

Abstraction by operation

"Computers are the only native speakers in the formal language of mathematics."

Right ?

So expose students to this *formal language*

- i.e. to speak by ***mechanical*** *operation*
- in trial & error learning
- with feed-back from TP-based systems.
- to foster *formal* abstraction (???)

A cause for future research !

Abstraction by operation

"Computers are the only native speakers in the formal language of mathematics."

Right ?

So expose students to this *formal language*

- i.e. to speak by ***mechanical*** *operation*
- in trial & error learning
- with feed-back from TP-based systems.
- to foster *formal* abstraction (???)

A cause for future research !

Abstraction by operation

"Computers are the only native speakers in the formal language of mathematics."

Right ?

So expose students to this *formal language*

- i.e. to speak by ***mechanical*** *operation*
- in trial & error learning
- with feed-back from TP-based systems.
- to foster *formal* abstraction (???)

A cause for future research !

Abstraction by operation

"Computers are the only native speakers in the formal language of mathematics."

Right ?

So expose students to this *formal language*

- i.e. to speak by ***mechanical*** *operation*
- in trial & error learning
- with feed-back from TP-based systems.
- to foster *formal* abstraction (???)

A cause for future research !

Non-formal Justifications . . .

. . . of the first three lines `2121, 2123, 2125` given by:

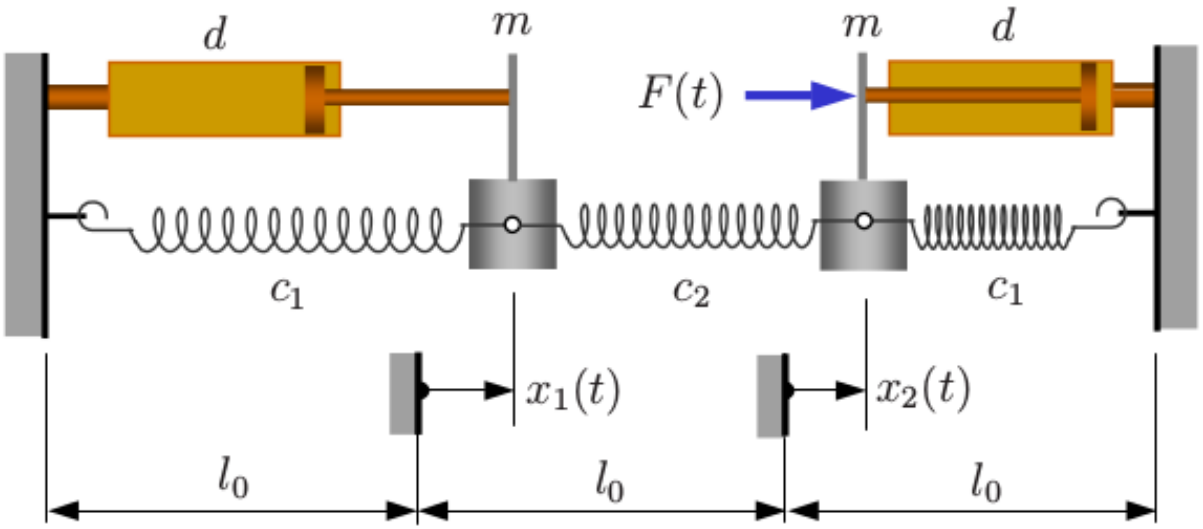


Figure : System with two oscillating masses ©W.Steiner 2015

Watch the demo at 09:30 !!!

21 `Problem` [determine, 2-mass-oscillator, DiffEq]:
211 `Specification`:
212 `Solution`:

2121 forces of springs

$$[F_{c1} = c_1 x_1,\ F_{c2} = c_2(x_2 - x_1),\ F_{c3} = c_1 x_2]$$

2122 forces of dampers

$$[F_{d1} = d\dot{x}_1,\ F_{d2} = d\dot{x}_2]$$

2123 mass times acceleration equals sum of all forces

$$[m\ddot{x}_1 = -F_{c1} + F_{c2} - F_{d1},\quad m\ddot{x}_2 = -F_{c2} - F_{c3} - F_{d2} + F_2]$$

2124 `Substitute` $[F_{c1}, F_{c2}, F_{c3}, F_{d1}, F_{d2}]$

$$[m\ddot{x}_1 = -c_1 x_1 + c_2(c_2 - x_1) - d\dot{x}_1,\quad m\ddot{x}_2 = -c_2(c_2 - x_1) - c_1 x_2 - d\dot{x}_2 + F_2]$$

2125 `Rewrite_Set` normalise

$$[m\ddot{x}_1 + d\dot{x}_1 + c_1 x_1 - c_2(x_2 - x_1) = 0,\quad m\ddot{x}_2 + d\dot{x}_2 + c_2(x_2 - x_1) + c_1 x_1 = F_2]$$

2126 switch to vector representation

$$\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{pmatrix} + \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} + \begin{pmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_1 + c_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ F \end{pmatrix}$$

22

$$\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \ddot{x} + \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} \dot{x} + \begin{pmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_1 + c_2 \end{pmatrix} x = \begin{pmatrix} 0 \\ F \end{pmatrix}$$

Outline

Cover all problem solving

Phases of problem solving:

1. **modeling**: translate example into formal specification
2. **specifying**: relate specification to problems, methods
3. **solving**: apply a method to stepwise solve a problem
4. go into sub-problems **recursively** . . .

TP-based software can be a

"complete", transparent and interactive

model of mathematics

Cover all problem solving

Phases of problem solving:

1. **modeling**: translate example into formal specification
2. **specifying**: relate specification to problems, methods
3. **solving**: apply a method to stepwise solve a problem
4. go into sub-problems **recursively** . . .

TP-based software can be a

"complete", transparent and interactive

model of mathematics

Cover all problem solving

Phases of problem solving:

1. **modeling**: translate example into formal specification
2. **specifying**: relate specification to problems, methods
3. **solving**: apply a method to stepwise solve a problem
4. go into sub-problems **recursively** . . .

TP-based software can be a

"complete", transparent and interactive

model of mathematics

Cover all problem solving

Phases of problem solving:

1. **modeling**: translate example into formal specification
2. **specifying**: relate specification to problems, methods
3. **solving**: apply a method to stepwise solve a problem
4. go into sub-problems **recursively** . . .

TP-based software can be a

"complete", transparent and interactive

model of mathematics

Cover all problem solving

Phases of problem solving:

1. **modeling**: translate example into formal specification
2. **specifying**: relate specification to problems, methods
3. **solving**: apply a method to stepwise solve a problem
4. go into sub-problems **recursively** . . .

TP-based software can be a

"complete", transparent and interactive

model of mathematics

Outline

Support independent learning

Why not in courses on applied mathematics let

1. independently study formal definitions, concrete algorithms, mechanical justifications, etc
2. in the lecture answer students' questions (which never should be left to machines)

as frequently done in courses on humanities ?

And **not**

1. lecture
2. exercise

as done since centuries

Support independent learning

Why not in courses on applied mathematics let

1. independently study formal definitions, concrete algorithms, mechanical justifications, etc
2. in the lecture answer students' questions (which never should be left to machines)

as frequently done in courses on humanities ?

And **not**

1. lecture
2. exercise

as done since centuries

Support independent learning

Why not in courses on applied mathematics let

1. independently study formal definitions, concrete algorithms, mechanical justifications, etc
2. in the lecture answer students' questions (which never should be left to machines)

as frequently done in courses on humanities ?

And **not**

1. lecture
2. exercise

as done since centuries

Outline

1. Misunderstandings ???
2. TP's benefits for educational math software
   - TP's knowledge is self-contained
   - each step is justified
   - . . . + computation: Lucas-Interpretation
   - TPs are open source
3. Promises of future software based on TP
   - Foster abstraction by operation
   - Cover all phases of problem solving
   - Support independent learning
   - Connect application — theory
4. Watch the demo at xx:xx !!!

Application — theory

at faculties of engineering:

theory	**application**
initial semesters	higher semesters
algebra, analysis, etc; integrate, differentiate, equ.solving	laboratories in physics, electronics, mechanics, etc
"What is this for?" (so skip as much as possible)	*"We never heard about!"* Time lost for repetition

theory . . . within *one* interactive exercise . . . application
(collections of exercises)

jump to integral, exercise, system does rest automatically. *probably* look at whole example	students review on their own focus on formal modeling, specification, methods

Application — theory

at faculties of engineering:

theory	**application**
initial semesters	higher semesters
algebra, analysis, etc; integrate, differentiate, equ.solving	laboratories in physics, electronics, mechanics, etc
"What is this for?" (so skip as much as possible)	*"We never heard about!"* Time lost for repetition

theory . . . within *one* interactive exercise . . . application
(collections of exercises)

jump to integral, exercise, system does rest automatically. *probably* look at whole example	students review on their own focus on formal modeling, specification, methods

Application — theory

at faculties of engineering:

theory	**application**
initial semesters	higher semesters
algebra, analysis, etc; integrate, differentiate, equ.solving	laboratories in physics, electronics, mechanics, etc
"What is this for?" (so skip as much as possible)	*"We never heard about!"* Time lost for repetition

theory . . . within *one* interactive exercise . . . application
(collections of exercises)

jump to integral, exercise, system does rest automatically. *probably* look at whole example	students review on their own focus on formal modeling, specification, methods

Application — theory

at faculties of engineering:

theory	**application**
initial semesters	higher semesters
algebra, analysis, etc; integrate, differentiate, equ.solving	laboratories in physics, electronics, mechanics, etc
"What is this for?" (so skip as much as possible)	*"We never heard about!"* Time lost for repetition

theory . . . within *one* interactive exercise . . . application

(collections of exercises)

jump to integral, exercise, system does rest automatically. *probably* look at whole example	students review on their own focus on formal modeling, specification, methods

Application — theory

at faculties of engineering:

theory	**application**
initial semesters	higher semesters
algebra, analysis, etc; integrate, differentiate, equ.solving	laboratories in physics, electronics, mechanics, etc
"What is this for?" (so skip as much as possible)	*"We never heard about!"* Time lost for repetition

theory . . . within *one* interactive exercise . . . application
(collections of exercises)

jump to integral, exercise, system does rest automatically. *probably* look at whole example	students review on their own focus on formal modeling, specification, methods

Application — theory

at faculties of engineering:

theory	**application**
initial semesters	higher semesters
algebra, analysis, etc; integrate, differentiate, equ.solving	laboratories in physics, electronics, mechanics, etc
"What is this for?" (so skip as much as possible)	*"We never heard about!"* Time lost for repetition

theory . . . within *one* interactive exercise . . . application
(collections of exercises)

jump to integral, exercise, system does rest automatically. *probably* look at whole example	students review on their own focus on formal modeling, specification, methods

Outline

1. Misunderstandings ???
2. TP's benefits for educational math software
   - TP's knowledge is self-contained
   - each step is justified
   - . . . + computation: Lucas-Interpretation
   - TPs are open source
3. Promises of future software based on TP
   - Foster abstraction by operation
   - Cover all phases of problem solving
   - Support independent learning
   - Connect application — theory
4. Watch the demo at xx:xx !!!

DEDUCTION: Systems that explain themselves

Walther Neuper