REPRESENTATION AND MANAGEMENT OF MATHEMATICS IN THEOREMA 2.0

Wolfgang Windsteiger
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University Linz (JKU)

Minisymposium From Information to Knowledge Management
Salzburg, September 12, 2017

JOHANNES KEPLER UNIVERSITÄT LINZ

RESEARCH INSTITUTE FOR SYMBOLIC COMPUTATION

THE THEOREMA SYSTEM: OVERVIEW

- Theorema is a mathematical assistant system aiming at communication with users in natural style.

THE THEOREMA SYSTEM: OVERVIEW

- Theorema is a mathematical assistant system aiming at communication with users in natural style.
- Theorema tries to support as many typical activities of a working mathematician as possible (computation & proving).

THE THEOREMA SYSTEM: OVERVIEW

- Theorema is a mathematical assistant system aiming at communication with users in natural style.
- Theorema tries to support as many typical activities of a working mathematician as possible (computation & proving).
- Natural style input: not only formulas but entire documents.

THE THEOREMA SYSTEM: OVERVIEW

- Theorema is a mathematical assistant system aiming at communication with users in natural style.
- Theorema tries to support as many typical activities of a working mathematician as possible (computation & proving).
- Natural style input: not only formulas but entire documents.
- General focus on automated proving.

THE THEOREMA SYSTEM: OVERVIEW

- Theorema is a mathematical assistant system aiming at communication with users in natural style.
- Theorema tries to support as many typical activities of a working mathematician as possible (computation & proving).
- Natural style input: not only formulas but entire documents.
- General focus on automated proving.
- Natural style output: generate proofs in human-readable style.

THE THEOREMA SYSTEM: OVERVIEW

- Theorema is a mathematical assistant system aiming at communication with users in natural style.
- Theorema tries to support as many typical activities of a working mathematician as possible (computation & proving).
- Natural style input: not only formulas but entire documents.
- General focus on automated proving.
- Natural style output: generate proofs in human-readable style.
- Developed on the basis of Mathematica, Theorema code is open source GPL-licensed.

THE THEOREMA SYSTEM: OVERVIEW

- Theorema is a mathematical assistant system aiming at communication with users in natural style.
- Theorema tries to support as many typical activities of a working mathematician as possible (computation & proving).
- Natural style input: not only formulas but entire documents.
- General focus on automated proving.
- Natural style output: generate proofs in human-readable style.
- Developed on the basis of Mathematica, Theorema code is open source GPL-licensed.
- User Interface: prepare mathematical document + support activities whose results get incorporated into the document.

EXAMPLE (USER INTERFACE)

Alice prepares lecture notes for Linear Algebra.

EXAMPLE (USER INTERFACE)

Alice prepares lecture notes for Linear Algebra.

She writes structured text (sections, subsections, etc.) into her mathematical document.

EXAMPLE (USER INTERFACE)

Alice prepares lecture notes for Linear Algebra.

She writes structured text (sections, subsections, etc.) into her mathematical document.

Text will also contain definitions and theorems (should appear in a familiar style).

EXAMPLE (USER INTERFACE)

Alice prepares lecture notes for Linear Algebra.

She writes structured text (sections, subsections, etc.) into her mathematical document.

Text will also contain definitions and theorems (should appear in a familiar style).

Document should contain example computations.

EXAMPLE (USER INTERFACE)

Alice prepares lecture notes for Linear Algebra.

She writes structured text (sections, subsections, etc.) into her mathematical document.

Text will also contain definitions and theorems (should appear in a familiar style).

Document should contain example computations.

Document should contain proofs.

EXAMPLE (USER INTERFACE)

Alice prepares lecture notes for Linear Algebra.

She writes structured text (sections, subsections, etc.) into her mathematical document.

Text will also contain definitions and theorems (should appear in a familiar style).

Document should contain example computations.

Document should contain proofs.

ALL IN ONE DOCUMENT

THEORY DEVELOPMENT

Theorema Theory = Theorema notebook + Theorema archive

Theorema notebook: mathematical document (see above).

THEORY DEVELOPMENT

Theorema Theory = Theorema notebook + Theorema archive

Theorema notebook: mathematical document (see above).

Theorema archive:

THEORY DEVELOPMENT

Theorema Theory = Theorema notebook + Theorema archive

Theorema notebook: mathematical document (see above).

Theorema archive:

- file containing formulas only

THEORY DEVELOPMENT

Theorema Theory = Theorema notebook + Theorema archive

Theorema notebook: mathematical document (see above).

Theorema archive:

- file containing formulas only
- in a format that can be imported into Theorema again

THEORY DEVELOPMENT

Theorema Theory = Theorema notebook + Theorema archive

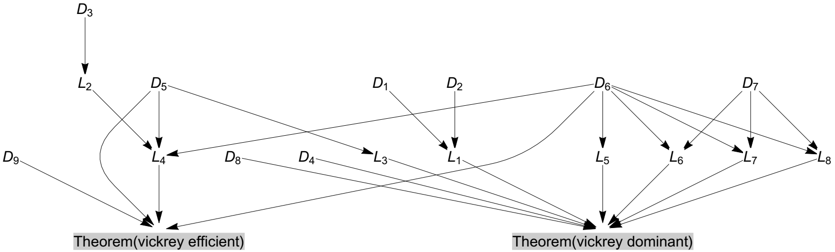
Theorema notebook: mathematical document (see above).

Theorema archive:

- file containing formulas only
- in a format that can be imported into Theorema again
- in other formats to be read by other systems, eg. MMT (Kohlhase/Rabe)

SUPPORT FOR THEORY DEVELOPMENT

Work in progress: Automatically maintain and display a formula graph such as



where

- vertices are formula labels and
- $X \to Y$ means: formula X is needed in the proof of formula Y.

SUPPORT FOR THEORY DEVELOPMENT

Features supported in the interactive graphical user interface (Theorema commander):

- Tooltip over label shows entire formula (see labels in proofs).

SUPPORT FOR THEORY DEVELOPMENT

Features supported in the interactive graphical user interface (Theorema commander):

- Tooltip over label shows entire formula (see labels in proofs).
- Color-coding of labels indicates theory membership.

SUPPORT FOR THEORY DEVELOPMENT

Features supported in the interactive graphical user interface (Theorema commander):

- Tooltip over label shows entire formula (see labels in proofs).
- Color-coding of labels indicates theory membership.
- Mouse-click on labels jumps to formula definition in the notebook.

SUPPORT FOR THEORY DEVELOPMENT

Features supported in the interactive graphical user interface (Theorema commander):

- Tooltip over label shows entire formula (see labels in proofs).
- Color-coding of labels indicates theory membership.
- Mouse-click on labels jumps to formula definition in the notebook.
- Zoom-out in order to switch to theory level, ie. vertices are theories and $X \to Y$ means: some formula of X is needed in the proof of some formula in Y.

SUPPORT FOR THEORY DEVELOPMENT

Features supported in the interactive graphical user interface (Theorema commander):

- Tooltip over label shows entire formula (see labels in proofs).
- Color-coding of labels indicates theory membership.
- Mouse-click on labels jumps to formula definition in the notebook.
- Zoom-out in order to switch to theory level, ie. vertices are theories and $X \to Y$ means: some formula of X is needed in the proof of some formula in Y.
- Show/Hide theories via checkboxes.

SUPPORT FOR THEORY DEVELOPMENT

Features supported in the interactive graphical user interface (Theorema commander):

- Tooltip over label shows entire formula (see labels in proofs).
- Color-coding of labels indicates theory membership.
- Mouse-click on labels jumps to formula definition in the notebook.
- Zoom-out in order to switch to theory level, ie. vertices are theories and $X \to Y$ means: some formula of X is needed in the proof of some formula in Y.
- Show/Hide theories via checkboxes.
- Apply graph algorithms in order to investigate theory structure.

FORMULA DEPENDENCIES

Prerequisite for dynamically maintaining the formula graph: tracking of formula dependencies.

FORMULA DEPENDENCIES

Prerequisite for dynamically maintaining the formula graph: tracking of formula dependencies.

Input to an automated prover: proof goal G, knowledge base $(k_1, \ldots, k_n)$, etc.

FORMULA DEPENDENCIES

Prerequisite for dynamically maintaining the formula graph: tracking of formula dependencies.

Input to an automated prover: proof goal G, knowledge base $(k_1, \ldots, k_n)$, etc.

Naive approach: Add edges $(k_1, G), \ldots, (k_n, G)$ to the formula graph.

FORMULA DEPENDENCIES

Prerequisite for dynamically maintaining the formula graph: tracking of formula dependencies.

Input to an automated prover: proof goal G, knowledge base $(k_1, \ldots, k_n)$, etc.

Naive approach: Add edges $(k_1, G), \ldots, (k_n, G)$ to the formula graph.

But: It is not guaranteed, that all of the $k_1, \ldots, k_n$ are really required for proving G.

FORMULA DEPENDENCIES

Prerequisite for dynamically maintaining the formula graph: tracking of formula dependencies.

Input to an automated prover: proof goal G, knowledge base $(k_1, \ldots, k_n)$, etc.

Naive approach: Add edges $(k_1, G), \ldots, (k_n, G)$ to the formula graph.

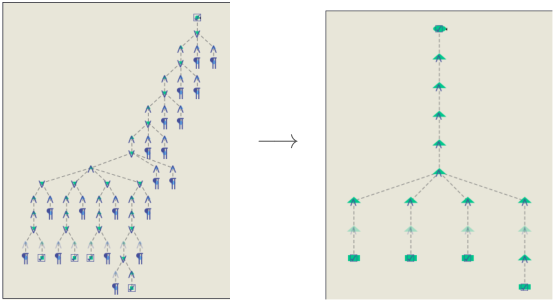
But: It is not guaranteed, that all of the $k_1, \ldots, k_n$ are really required for proving G.

Solution:

PROOF SIMPLIFICATION.

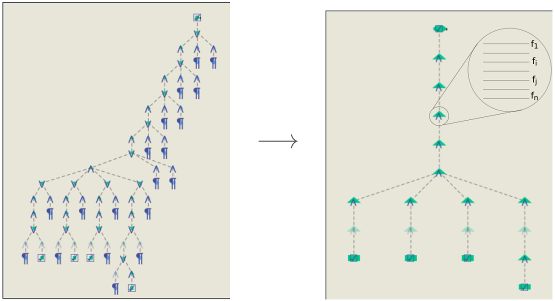
PROOF SIMPLIFICATION

1. Eliminate unnecessary paths in the proof tree



PROOF SIMPLIFICATION

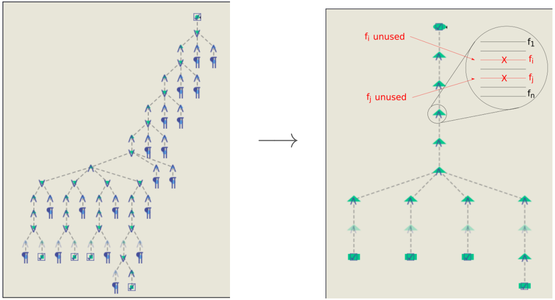
1. Eliminate unnecessary paths in the proof tree
2. Eliminate unnecessary formulas in each node



RISC RESEARCH INSTITUTE FOR SYMBOLIC COMPUTATION

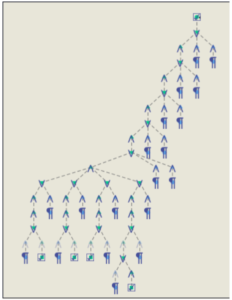
PROOF SIMPLIFICATION

1. Eliminate unnecessary paths in the proof tree
2. Eliminate unnecessary formulas in each node

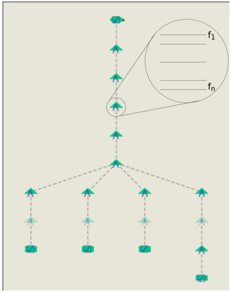


PROOF SIMPLIFICATION

1. Eliminate unnecessary paths in the proof tree
2. Eliminate unnecessary formulas in each node



$\longrightarrow$



DETECTION OF UNNECESSARY FORMULAS

Proof tree node: proof situation (proof goal + knowledge base + additional prover info)

DETECTION OF UNNECESSARY FORMULAS

Proof tree node: proof situation (proof goal + knowledge base + additional prover info)

Proof tree node (detail): proof info + proof situation

DETECTION OF UNNECESSARY FORMULAS

Proof tree node: proof situation (proof goal + knowledge base + additional prover info)

Proof tree node (detail): proof info + proof situation

Proof info: contains information necessary for displaying proof in human-readable form, eg. formulas needed to perform the proof step.

DETECTION OF UNNECESSARY FORMULAS

Proof tree node: proof situation (proof goal + knowledge base + additional prover info)

Proof tree node (detail): proof info + proof situation

Proof info: contains information necessary for displaying proof in human-readable form, eg. formulas needed to perform the proof step.

Proof info cannot be extracted automatically, it has to be specified when implementing the prover.

PROOF INFO: EXAMPLE

Proof Step: Reduce proof goal g by knowledge k_i to new goal g'.

PROOF INFO: EXAMPLE

Proof Step: Reduce proof goal g by knowledge k_i to new goal g'.

Node	Proof Situation	Proof Info
N	$g, (k_1, \ldots, k_i, \ldots, k_n)$	
$\downarrow$	$\downarrow$	$\ldots\ldots\ldots\ldots$
child of N	$g', (k_1, \ldots, k_i, \ldots, k_n)$	

PROOF INFO: EXAMPLE

Proof Step: Reduce proof goal g by knowledge k_i to new goal g'.

Node	Proof Situation	Proof Info
N	$\boxed{g}, (k_1, \ldots, \boxed{k_i}, \ldots, k_n)$	
$\downarrow$	$\downarrow$	....
child of N	$\boxed{g'}, (k_1, \ldots, k_i, \ldots, k_n)$	Formulas needed: g, k_i, g'

PROOF INFO: EXAMPLE

Proof Step: Reduce proof goal g by knowledge k_i to new goal g'.

Node	Proof Situation	Proof Info
N	$g, (k_1, \ldots, k_i, \ldots, k_n)$	
$\downarrow$	$\downarrow$	....
child of N	$g', (k_1, \ldots, k_i, \ldots, k_n)$	Formulas needed: g, k_i, g'

Proof info allows to generate text:

"In order to prove g, because of k_i, it is sufficient to prove g'."

PROOF INFO: FORMULAS NEEDED

Instead of plain list $(f_1, \ldots, f_n)$

PROOF INFO: FORMULAS NEEDED

Instead of plain list $(f_1, \ldots, f_n)$

$$\longrightarrow \qquad ((u_1, \ldots, u_m), (g_1, \ldots, g_m))$$

where

u_i are sets of formulas used and

g_i are sets of formulas generated in a node, st.

all formulas in u_i are needed to generate the formulas in g_i.

PROOF INFO: FORMULAS NEEDED

Instead of plain list $(f_1, \dots, f_n)$

$$\longrightarrow \qquad ((u_1, \dots, u_m), (g_1, \dots, g_m))$$

where

u_i are sets of formulas used and

g_i are sets of formulas generated in a node, st.

all formulas in u_i are needed to generate the formulas in g_i.

Example above: $((\{g, k_i\}), (\{g'\}))$.

DELETING UNNECESSARY FORMULAS

Node	Proof Info	Necessary Formulas
Terminal Node	$((u_1), (\{\}))$	$F = u_1$

DELETING UNNECESSARY FORMULAS

Node	Proof Info	Necessary Formulas
N	$((u_1, \ldots, u_m), (g_1, \ldots, g_m))$	
$\downarrow$		
Terminal Node	$((u_1), (\{\}))$	$F = u_1$

DELETING UNNECESSARY FORMULAS

Node	Proof Info	Necessary Formulas
N	$((u'_1, \ldots, u'_m), (g'_1, \ldots, g'_m))$	
$\downarrow$		
Terminal Node	$((u_1), (\{\}))$	$F = u_1$

where

$$g'_i = g_i \cap F \qquad u'_i = \begin{cases} u_i & \text{if } g'_i \neq \emptyset \\ \emptyset & \text{if } g'_i = \emptyset \end{cases}$$

DELETING UNNECESSARY FORMULAS

Node	Proof Info	Necessary Formulas
N	$((u'_1, \dots, u'_m), (g'_1, \dots, g'_m))$	$F = F \cup u'_1 \cup \dots \cup u'_m$
$\downarrow$		
Terminal Node	$((u_1), (\{\}))$	$F = u_1$

DELETING UNNECESSARY FORMULAS

Node	Proof Info	Necessary Formulas
parent of N	$((u_1, \ldots, u_m), (g_1, \ldots, g_m))$	
$\downarrow$		
N	$((u'_1, \ldots, u'_m), (g'_1, \ldots, g'_m))$	$F = F \cup u'_1 \cup \cdots \cup u'_m$
$\downarrow$		
Terminal Node	$((u_1), (\{\}))$	$F = u_1$

DELETING UNNECESSARY FORMULAS

Node	Proof Info	Necessary Formulas
parent of N	$((u_1^*, \ldots, u_m^*), (g_1^*, \ldots, g_m^*))$	
$\downarrow$		
N	$((u_1', \ldots, u_m'), (g_1', \ldots, g_m'))$	$F = F \cup u_1' \cup \cdots \cup u_m'$
$\downarrow$		
Terminal Node	$((u_1), (\{\}))$	$F = u_1$

where

$$g_i^* = g_i \cap F \qquad u_i^* = \begin{cases} u_i & \text{if } g_i^* \neq \emptyset \\ \emptyset & \text{if } g_i^* = \emptyset \end{cases}$$

DELETING UNNECESSARY FORMULAS

Node	Proof Info	Necessary Formulas
parent of N	$((u_1^*, \ldots, u_m^*), (g_1^*, \ldots, g_m^*))$	$F = F \cup u_1^* \cup \cdots \cup u_m^*$
$\downarrow$		
N	$((u_1', \ldots, u_m'), (g_1', \ldots, g_m'))$	$F = F \cup u_1' \cup \cdots \cup u_m'$
$\downarrow$		
Terminal Node	$((u_1), (\{\}))$	$F = u_1$

DELETING UNNECESSARY FORMULAS

Node	Proof Info	Necessary Formulas
parent of N	$((u_1^*, \ldots, u_m^*), (g_1^*, \ldots, g_m^*))$	$F = F \cup u_1^* \cup \cdots \cup u_m^*$
$\downarrow$		
N	$((u_1', \ldots, u_m'), (g_1', \ldots, g_m'))$	$F = F \cup u_1' \cup \cdots \cup u_m'$
$\downarrow$		
Terminal Node	$((u_1), (\{\}))$	$F = u_1$

Upon termination: F contains all necessary formulas!

Root node proof info: $((\{\}), (K^*))$ where K^* contains just those formulas of the original knowledge base that were actually needed.

DEMO: PROOF SIMPLIFICATION

Lemma (Winner's Payment)

$$\forall_{b,x,p,i}$$

If b_i wins and also $b_{i\leftarrow a}$ wins, then i pays the same amount because the maximum of the others did not change:

$$\texttt{secondPriceAuctionWinner}[b, x, p, i] \Rightarrow \forall_{y,q,a} \left(\texttt{secondPriceAuctionWinner}[b_{i\leftarrow a}, y, q, i] \Rightarrow p_i == q_i \wedge x_i == 1 \wedge y_i == 1\right)$$

(win – win)

If b_i wins and $b_{i\leftarrow a}$ loses, then a is not the maximum bid, hence the maximum of the rest must be greater or equal to a:

$$\texttt{secondPriceAuctionWinner}[b, x, p, i] \Rightarrow \forall_{y,q,a} \left(\texttt{secondPriceAuctionLoser}[b_{i\leftarrow a}, y, q, i] \Rightarrow q_i == 0 \wedge x_i == 1 \wedge y_i == 0 \wedge 0 \geq a - p_i\right)$$

(win – lose)

If b_i loses and $b_{i\leftarrow a}$ wins, then a is the maximum bid and, hence, greater or equal the payment, because this is the maximum of the rest:

$$\texttt{secondPriceAuctionLoser}[b, x, p, i] \Rightarrow \forall_{y,q,a} \left(\texttt{secondPriceAuctionWinner}[b_{i\leftarrow a}, y, q, i] \Rightarrow p_i == 0 \wedge x_i == 0 \wedge y_i == 1 \wedge a - q_i \geq 0\right)$$

(lose – win)

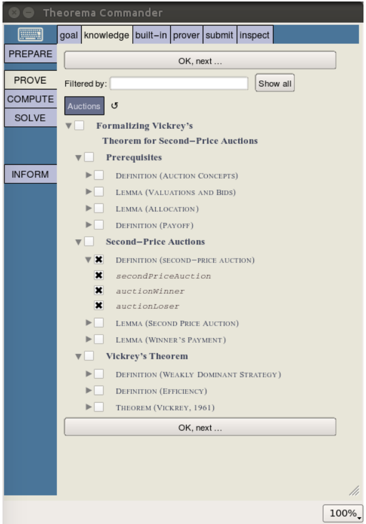
If b_i loses and also $b_{i\leftarrow a}$ loses, then i does not get anything and has to pay nothing:

$$\texttt{secondPriceAuctionLoser}[b, x, p, i] \Rightarrow \forall_{y,q,a} \left(\texttt{secondPriceAuctionLoser}[b_{i\leftarrow a}, y, q, i] \Rightarrow p_i == 0 \wedge x_i == 0 \wedge y_i == 0 \wedge q_i == 0\right)$$

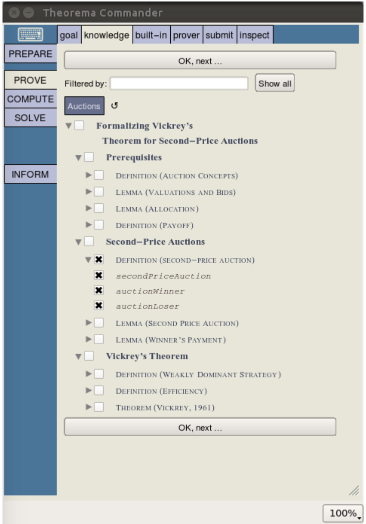
(lose – lose)

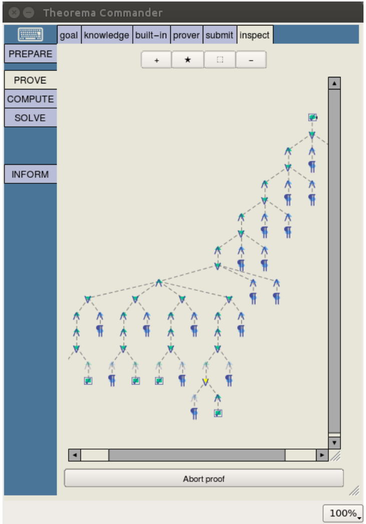
RISC
RESEARCH INSTITUTE FOR SYMBOLIC COMPUTATION

DEMO: PROOF SIMPLIFICATION



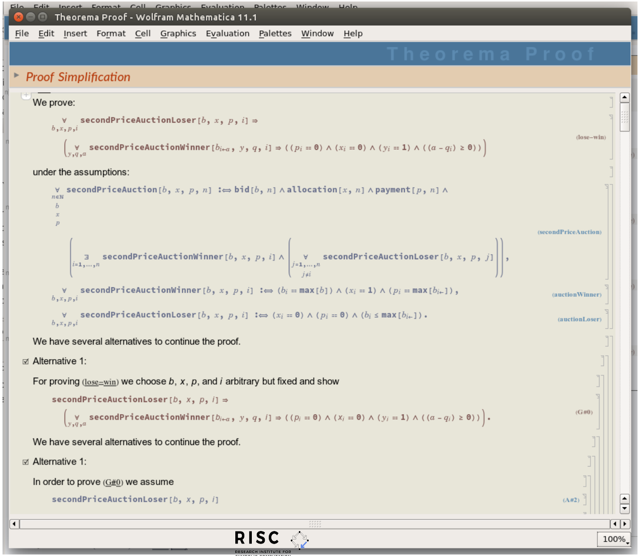
DEMO: PROOF SIMPLIFICATION



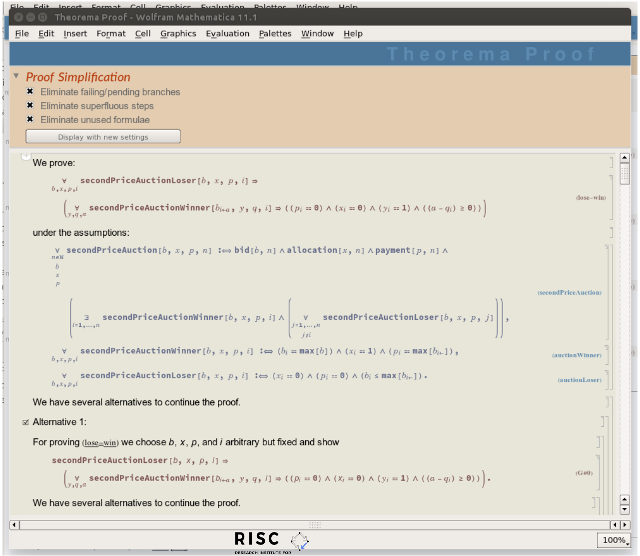


DEMO: PROOF SIMPLIFICATION

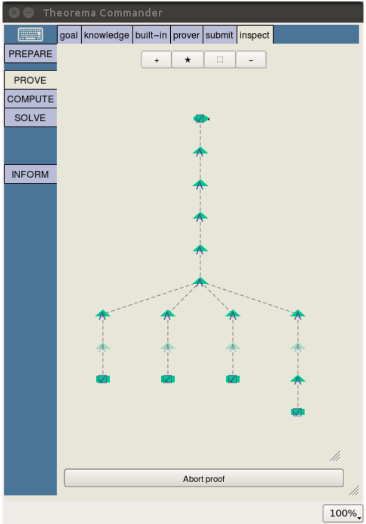
DEMO: PROOF SIMPLIFICATION



DEMO: PROOF SIMPLIFICATION



DEMO: PROOF SIMPLIFICATION



DEMO: PROOF SIMPLIFICATION

JKU

RISC
RESEARCH INSTITUTE FOR SYMBOLIC COMPUTATION

CONCLUSION

- Theory exploration/development is an important aspect of knowledge/information management.

CONCLUSION

- Theory exploration/development is an important aspect of knowledge/information management.
- Formula dependency graph is a nice/valuable tool for theory exploration/development.

CONCLUSION

- Theory exploration/development is an important aspect of knowledge/information management.
- Formula dependency graph is a nice/valuable tool for theory exploration/development.
- Proof simplification is a necessary requirement, if we want to automatically maintain a formula dependency graph.