

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

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

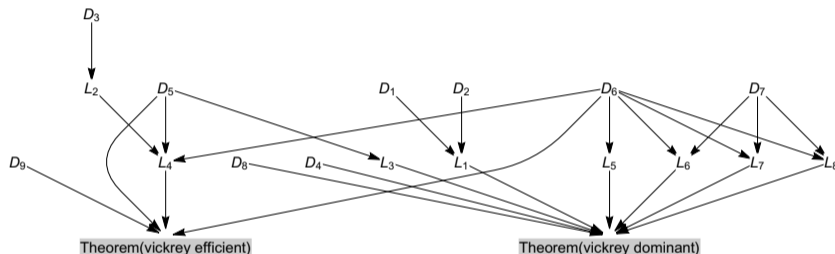
Theorema notebook: mathematical document (see above).

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
- $X \rightarrow Y$ 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 $X \rightarrow Y$ 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 $X \rightarrow Y$ 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 $X \rightarrow Y$ 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, \dots, 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, \dots, k_n) , etc.

Naive approach: Add edges $(k_1, G), \dots, (k_n, G)$ 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, \dots, k_n) , etc.

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

But: It is not guaranteed, that all of the k_1, \dots, 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, \dots, k_n) , etc.

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

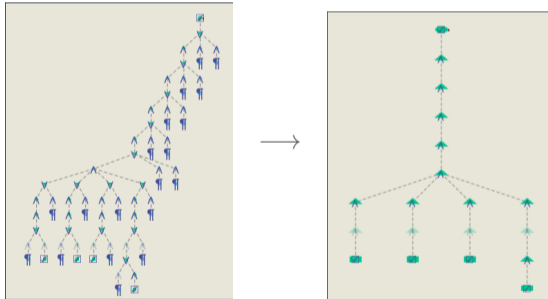
But: It is not guaranteed, that all of the k_1, \dots, k_n are really required for proving G .

Solution:

PROOF SIMPLIFICATION.

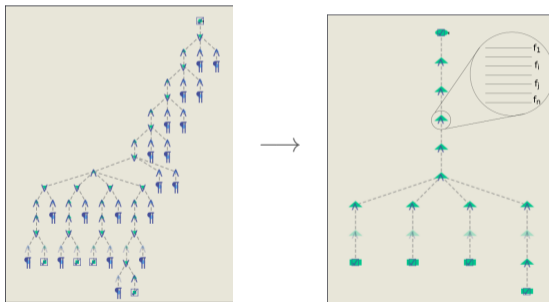
PROOF SIMPLIFICATION

1. Eliminate unnecessary paths in the proof tree



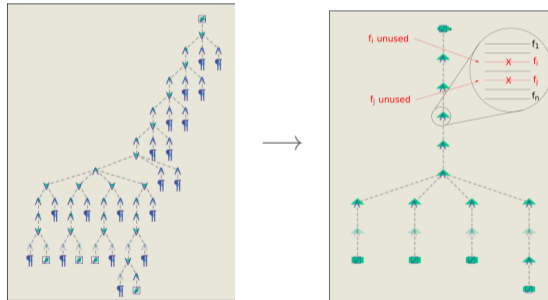
PROOF SIMPLIFICATION

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



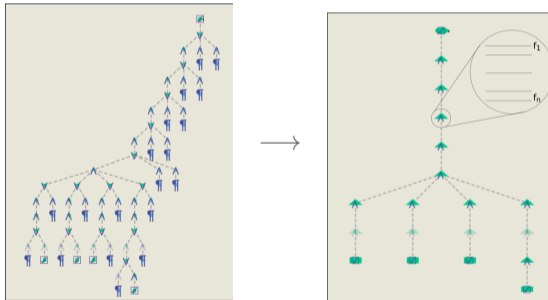
PROOF SIMPLIFICATION

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



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	$g, (k_1, \dots, k_i, \dots, k_n)$	
↓	↓
child of N	$g', (k_1, \dots, k_i, \dots, k_n)$	

PROOF INFO: EXAMPLE

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

Node	Proof Situation	Proof Info
N	$\boxed{g}, (k_1, \dots, \boxed{k_i}, \dots, k_n)$
\downarrow	\downarrow	
child of N	$\boxed{g'}, (k_1, \dots, k_i, \dots, k_n)$	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	$g, (k_1, \dots, k_i, \dots, k_n)$	
\downarrow	\downarrow
child of N	$g', (k_1, \dots, k_i, \dots, k_n)$	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, \dots, f_n)

PROOF INFO: FORMULAS NEEDED

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

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

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 .

PROOF INFO: FORMULAS NEEDED

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

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

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	$((u_1, \dots, u_m), (g_1, \dots, g_m))$	
\downarrow		
Terminal Node	$((u_1), (\{\}))$	$F = u_1$

DELETING UNNECESSARY FORMULAS

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

where

$$g'_i = g_i \cap F$$

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

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	$((u_1, \dots, u_m), (g_1, \dots, g_m))$	
↓		
N	$((u'_1, \dots, u'_m), (g'_1, \dots, g'_m))$	$F = F \cup u'_1 \cup \dots \cup u'_m$
↓		
Terminal Node	$((u_1), (\{\}))$	$F = u_1$

DELETING UNNECESSARY FORMULAS

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

DELETING UNNECESSARY FORMULAS

Node	Proof Info	Necessary Formulas
parent of N	$((u_1^*, \dots, u_m^*), (g_1^*, \dots, g_m^*))$	$F = F \cup u_1^* \cup \dots \cup u_m^*$
↓		
N	$((u_1', \dots, u_m'), (g_1', \dots, g_m'))$	$F = F \cup u_1' \cup \dots \cup u_m'$
↓		
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-a} wins, then i pays the same amount because the maximum of the others did not change:

$$\text{secondPriceAuctionWinner}[b, x, p, i] \Rightarrow$$
$$\forall y, q, a \quad (\text{secondPriceAuctionWinner}[b_{i-a}, y, q, i] \Rightarrow p_i = q_i \wedge x_i = 1 \wedge y_i = 1)$$

(win - win) ✕

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

$$\text{secondPriceAuctionWinner}[b, x, p, i] \Rightarrow$$
$$\forall y, q, a \quad (\text{secondPriceAuctionLoser}[b_{i-a}, y, q, i] \Rightarrow q_i = 0 \wedge x_i = 1 \wedge y_i = 0 \wedge 0 \geq a - p_i)$$

(win - lose) ✕

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

$$\text{secondPriceAuctionLoser}[b, x, p, i] \Rightarrow$$
$$\forall y, q, a \quad (\text{secondPriceAuctionWinner}[b_{i-a}, y, q, i] \Rightarrow p_i = 0 \wedge x_i = 0 \wedge y_i = 1 \wedge a - q_i \geq 0)$$

(lose - win) ✕

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

$$\text{secondPriceAuctionLoser}[b, x, p, i] \Rightarrow$$
$$\forall y, q, a \quad (\text{secondPriceAuctionLoser}[b_{i-a}, y, q, i] \Rightarrow p_i = 0 \wedge x_i = 0 \wedge y_i = 0 \wedge q_i = 0)$$

(lose - lose) ✕

DEMO: PROOF SIMPLIFICATION

The screenshot shows the 'Theorema Commander' application window. The interface includes a top menu bar with buttons for 'goal', 'knowledge', 'built-in', 'prover', 'submit', and 'inspect'. On the left, a vertical sidebar contains buttons for 'PREPARE', 'PROVE', 'COMPUTE', 'SOLVE', and 'INFORM'. The main workspace displays a hierarchical tree structure under the heading 'Auctions'. The tree is expanded to show 'Formalizing Vickrey's Theorem for Second-Price Auctions', which includes 'Prerequisites' and 'Second-Price Auctions'. Under 'Second-Price Auctions', the following items are listed: 'DEFINITION (SECOND-PRICE AUCTION)' (expanded), 'secondPriceAuction', 'auctionWinner', and 'auctionLoser' (all with an 'x' icon), 'LEMMA (SECOND PRICE AUCTION)', and 'LEMMA (WINNER'S PAYMENT)'. Under 'Vickrey's Theorem', the items are 'DEFINITION (WEAKLY DOMINANT STRATEGY)', 'DEFINITION (EFFICIENCY)', and 'THEOREM (VICKREY, 1961)'. Buttons for 'OK, next ...' are present at the top and bottom of the main workspace. A '100%' zoom level indicator is visible in the bottom right corner.

DEMO: PROOF SIMPLIFICATION

The screenshot shows the 'Theorema Commander' interface. The top navigation bar includes 'goal', 'knowledge', 'built-in', 'prover', 'submit', and 'inspect'. The left sidebar has buttons for 'PREPARE', 'PROVE', 'COMPUTE', 'SOLVE', and 'INFORM'. The main area displays a tree of formalized concepts under the heading 'Formalizing Vickrey's Theorem for Second-Price Auctions'. The tree is expanded to show 'Prerequisites' and 'Second-Price Auctions'. Under 'Second-Price Auctions', several items are checked with an 'x' icon: 'DEFINITION (SECOND-PRICE AUCTION)', 'secondPriceAuction', 'auctionWinner', and 'auctionLoser'. Other items like 'LEMMA (SECOND PRICE AUCTION)' and 'LEMMA (WINNER'S PAYMENT)' are unchecked. A 'Vickrey's Theorem' section is also visible with unchecked items. A 'Show all' button is present next to the 'Filtered by:' field. At the bottom right, there is a '100%' zoom indicator.

The screenshot shows the 'Theorema Commander' interface with a large, complex proof tree displayed in the main area. The tree is a dense network of nodes connected by dashed lines, representing a complex proof structure. The nodes are represented by small icons, some of which are blue and some are green. The tree is oriented vertically, with the root at the top and branches extending downwards. The interface elements are the same as in the previous screenshot, including the top navigation bar and the left sidebar. At the bottom right, there is a '100%' zoom indicator and an 'Abort proof' button.

DEMO: PROOF SIMPLIFICATION

▼ Proof of (lose-win) #4: [Show proof](#)

knowledge built-in prover Restore settings

secondPriceAuction , auctionWinner , auctionLoser

DEMO: PROOF SIMPLIFICATION

File Edit Insert Format Cell Graphics Evaluation Palettes Window Help

Theorema Proof - Wolfram Mathematica 11.1

Theorema Proof

► Proof Simplification

We prove:

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

(lose-win)

under the assumptions:

$$\forall_{n \in \mathbb{N}} \text{secondPriceAuction}[b, x, p, n] \Rightarrow \text{bid}[b, n] \wedge \text{allocation}[x, n] \wedge \text{payment}[p, n] \wedge$$

$$b$$

$$x$$

$$p$$

(secondPriceAuction)

$$\left(\exists_{i \neq 1, \dots, n} \text{secondPriceAuctionWinner}[b, x, p, i] \wedge \left(\forall_{j \neq 1, \dots, n} \text{secondPriceAuctionLoser}[b, x, p, j] \right) \right),$$

$$\forall_{b,x,p,i} \text{secondPriceAuctionWinner}[b, x, p, i] \Rightarrow (b_i = \max[b]) \wedge (x_i = 1) \wedge (p_i = \max[b_{i-}]),$$

(auctionWinner)

$$\forall_{b,x,p,i} \text{secondPriceAuctionLoser}[b, x, p, i] \Rightarrow (x_i = 0) \wedge (p_i = 0) \wedge (b_i \leq \max[b_{i-}]).$$

(auctionLoser)

We have several alternatives to continue the proof.

Alternative 1:

For proving (lose-win) we choose $b, x, p,$ and i arbitrary but fixed and show

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

(G#0)

We have several alternatives to continue the proof.

Alternative 1:

In order to prove (G#0) we assume

$$\text{secondPriceAuctionLoser}[b, x, p, i]$$

(AK2)

RISC

100%

DEMO: PROOF SIMPLIFICATION

The screenshot shows the Wolfram Mathematica 11.1 interface with a custom 'Theorema Proof' window. The window title is 'Theorema Proof - Wolfram Mathematica 11.1'. The menu bar includes 'File', 'Edit', 'Insert', 'Format', 'Cell', 'Graphics', 'Evaluation', 'Palettes', 'Window', and 'Help'. The main content area is titled 'Theorema Proof' and contains a 'Proof Simplification' section with three checkboxes: 'Eliminate failing/pending branches', 'Eliminate superfluous steps', and 'Eliminate unused formulae'. A 'Display with new settings' button is located below these options.

The main proof area displays the following content:

We prove:

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

(loss-win)

under the assumptions:

$$\forall_{n \in \mathbb{N}} \text{secondPriceAuction}[b, x, p, n] \Leftrightarrow \text{bid}[b, n] \wedge \text{allocation}[x, n] \wedge \text{payment}[p, n] \wedge$$

(secondPriceAuction)

$$\left(\exists_{i \in \{1, \dots, n\}} \text{secondPriceAuctionWinner}[b, x, p, i] \wedge \forall_{j \neq i} \text{secondPriceAuctionLoser}[b, x, p, j] \right),$$
$$\forall_{b, x, p, i} \text{secondPriceAuctionWinner}[b, x, p, i] \Rightarrow (b_i = \max[b]) \wedge (x_i = 1) \wedge (p_i = \max[b_{i-}]),$$

(auctionWinner)

$$\forall_{b, x, p, i} \text{secondPriceAuctionLoser}[b, x, p, i] \Rightarrow (x_i = 0) \wedge (p_i = 0) \wedge (b_i \leq \max[b_{i-}]).$$

(auctionLoser)

We have several alternatives to continue the proof.

Alternative 1:

For proving (loss-win) we choose b , x , p , and i arbitrary but fixed and show

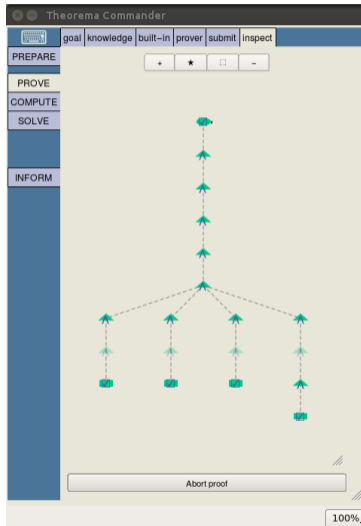
$$\text{secondPriceAuctionLoser}[b, x, p, i] \Rightarrow \left(\forall_{y, q, j} \text{secondPriceAuctionWinner}[b_{i \rightarrow j}, y, q, j] \Rightarrow ((p_i = 0) \wedge (x_i = 0) \wedge (y_i = 1) \wedge ((a - q_i) \geq 0)) \right).$$

(G#0)

We have several alternatives to continue the proof.

The bottom of the window shows the RISC logo (Research Institute for Symbolic Computation) and a 100% zoom level.

DEMO: PROOF SIMPLIFICATION



DEMO: PROOF SIMPLIFICATION

The screenshot displays the 'Theorema Proof' interface in Wolfram Mathematica 11.1. The window title is 'Theorema Proof - Wolfram Mathematica 11.1'. The menu bar includes 'File', 'Edit', 'Insert', 'Format', 'Cell', 'Graphics', 'Evaluation', 'Palettes', 'Window', and 'Help'. The main content area is titled 'Theorema Proof' and shows a 'Proof Simplification' section with a star icon and the text 'Time spent for simplifying the proof: 0.018791s'.

We prove:

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

under the assumptions:

$$\forall_{b, x, p, i} \text{secondPriceAuctionWinner}[b, x, p, i] := \text{(auctionWinner)}$$
$$(b_i = \max[b]) \wedge (x_i = 1) \wedge (p_i = \max[b_{i-}]) ,$$
$$\forall_{b, x, p, i} \text{secondPriceAuctionLoser}[b, x, p, i] := \text{(auctionLoser)}$$
$$(x_i = 0) \wedge (p_i = 0) \wedge (b_i \leq \max[b_{i-}]) .$$

For proving (lose-win) we choose b, x, p , and i arbitrary but fixed and show

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

In order to prove (G#0) we assume

$$\text{secondPriceAuctionLoser}[b, x, p, i] \text{ (A#2)}$$

and then prove

$$\forall_{y, q, a} \text{secondPriceAuctionWinner}[b_{i-a}, y, q, i] \Rightarrow \text{(G#3)}$$
$$((p_i = 0) \wedge (x_i = 0) \wedge (y_i = 1) \wedge ((a - q_i) \geq 0)) .$$

For proving (G#3) we choose y, q , and a arbitrary but fixed and show

$$\text{secondPriceAuctionWinner}[b_{i-a}, y, q, i] \Rightarrow ((p_i = 0) \wedge (x_i = 0) \wedge (y_i = 1) \wedge ((a - q_i) \geq 0)) . \text{(G#7)}$$

In order to prove (G#7) we assume

$$\text{secondPriceAuctionWinner}[b_{i-a}, y, q, i] \text{ (A#11)}$$

and then prove

$$(p_i = 0) \wedge (x_i = 0) \wedge (y_i = 1) \wedge ((a - q_i) \geq 0) . \text{(G#12)}$$

The interface also features a 'RISC' logo at the bottom center, which stands for the Research Institute for Symbolic Computation. The status bar at the bottom right shows '100%' zoom.

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.