Matita's User Interaction

Enrico Tassi

Microsoft Research-INRIA Joint Center

Journée ADT Interfaces — 27 October 2010 — Paris

History

Input

- Ambiguity support
- Tinycals
- UTF-8 support

Output

- MathML & friends
- Proof rendering
- GtkMathView
- Graphs

Metadata

- What's interesting about formal proofs?

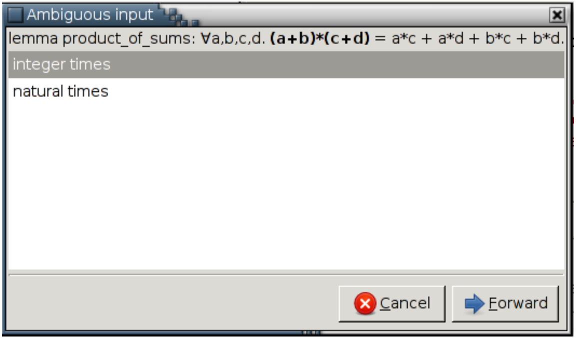
History

Matita was born from a rib of the MoWGLI project (Coq's library on the web)

- Web standards:
  - XML for CIC terms
  - MathML for content/presentation
  - CicBrowser (for the library)
- Natural language presentation of proof terms
- Ambiguity manager
- Searching facilities

Ambiguity manager

Operators and names can be overloaded. The intended interpretation if chosen among the valid ones interactively.

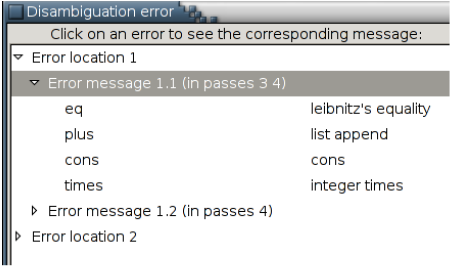


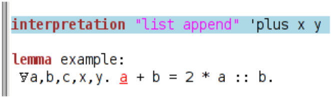
User's preferences are recorded in the script, and kept into account when interpreting the following commands.

Ambiguity manager: errors

Multiple interpretation also means multiple errors:

- Error messages must be equipped with the interpretation that generated them
- Spurious errors must be hidden
  - Many notions of “spurious”
  - The implemented one: located in a sub-formula that admits a valid interpretation





Tinycals: history

Original aim: make proof structuring/refactoring less painful.

Tinycals: history

Original aim: make proof structuring/refactoring less painful.

```
theorem associative_Ztimes:  associative  Z Ztimes.
unfold  associative .
 intros .
elim  x.
 simplify .
  reflexivity  .
elim  y.
 simplify .
  reflexivity  .
elim  z.
 simplify .
  reflexivity  .
change with
(pos (pred  ((S (pred  ((S n)  * (S n1))))  * (S n2)))  =
pos (pred  ((S n)  * (S (pred  ((S n1)  * (S n2 ))))))).
 rewrite  < S_pred.
 rewrite  < S_pred.
 rewrite  < assoc_times.
  reflexivity  .
apply  lt_O_times_S_S .
apply  lt_O_times_S_S .
change with
(neg (pred  ((S (pred  ((S n)  * (S n1))))  * (S n2)))  =
neg (pred  ((S n)  * (S (pred  ((S n1)  * (S n2 ))))))).
 rewrite  < S_pred.
 rewrite  < S_pred.
 rewrite  < assoc_times.
  reflexivity  .
```

```
apply  lt_O_times_S_S .
apply  lt_O_times_S_S .
elim  z.
 simplify .
  reflexivity  .
change with
(neg (pred  ((S (pred  ((S n)  * (S n1))))  * (S n2)))  =
neg (pred  ((S n)  * (S (pred  ((S n1)  * (S n2 ))))))).
 rewrite  < S_pred.
 rewrite  < S_pred.
 rewrite  < assoc_times.
  reflexivity  .
apply  lt_O_times_S_S .
apply  lt_O_times_S_S .
change with
(pos (pred  ((S (pred  ((S n)  * (S n1))))  * (S n2)))  =
pos(pred  ((S n)  * (S (pred  ((S n1)  * (S n2 ))))))).
 rewrite  < S_pred.
 rewrite  < S_pred.
 rewrite  < assoc_times.
  reflexivity  .
apply  lt_O_times_S_S .
apply  lt_O_times_S_S .
elim  y.
 simplify .
  reflexivity  .
elim  z.
. . .
```

Tinycals: what about indentation?

Indentation looks like a cheap solution

```
theorem associative_Ztimes: associative Z Ztimes.
unfold associative . intros .elim x.
  simplify . reflexivity .
  elim y.
    simplify . reflexivity .
    elim z.
      simplify . reflexivity .
      change with
       (pos (pred ((S (pred ((S n) * (S n1)))) * (S n2))) =
       pos (pred ((S n) * (S (pred ((S n1) * (S n2 ))))))).
        rewrite < S_pred.rewrite < S_pred.rewrite < assoc_times. reflexivity .
         apply lt_O_times_S_S .apply lt_O_times_S_S .
      change with
       (neg (pred ((S (pred ((S n) * (S n1)))) * (S n2))) =
       neg (pred ((S n) * (S (pred ((S n1) * (S n2 ))))))).
        rewrite < S_pred.rewrite < S_pred.rewrite < assoc_times. reflexivity .
         apply lt_O_times_S_S .apply lt_O_times_S_S .
    elim z.
      simplify . reflexivity .
      change with
       (neg (pred ((S (pred ((S n) * (S n1)))) * (S n2))) =
       neg (pred ((S n) * (S (pred ((S n1) * (S n2 ))))))).
        rewrite < S_pred.rewrite < S_pred.rewrite < assoc_times. reflexivity .
         apply lt_O_times_S_S .apply lt_O_times_S_S .
      change with
       (pos (pred ((S (pred ((S n) * (S n1)))) * (S n2))) =
       pos(pred ((S n) * (S (pred ((S n1) * (S n2 ))))))).
        rewrite < S_pred.rewrite < S_pred.rewrite < assoc_times. reflexivity .
        apply lt_O_times_S_S .apply lt_O_times_S_S .
  elim y.
...
```

Tinycals

Indentation only "suggests" the structure of a proof

- but it's not checked by the system

Why there were no tacticals?

- Hard to build a huge proof in one go with the executed=locked interaction style
- We are lazy, refactoring costs time
- Read a proof made with tacticals is harder

Tinycals

Indentation only "suggests" the structure of a proof

- but it's not checked by the system

Why there were no tacticals?

- Hard to build a huge proof in one go with the executed=locked interaction style
- We are lazy, refactoring costs time
- Read a proof made with tacticals is harder

The trick

- De-structured syntax
  - NO: $\langle T \rangle$ ::= "**[**" $\langle T \rangle$ "**|**" $\langle T \rangle$ "**]**" | ...
  - YES: $\langle T \rangle$ ::= "**[**" | "**|**" | "**]**" | ...

Tinycals

Indentation only "suggests" the structure of a proof

- but it's not checked by the system

Why there were no tacticals?

- Hard to build a huge proof in one go with the executed=locked interaction style
- We are lazy, refactoring costs time
- Read a proof made with tacticals is harder

The trick

- De-structured syntax
  - NO: $\langle T \rangle$ ::= "**[**" $\langle T \rangle$ "**|**" $\langle T \rangle$ "**]**" | ...
  - YES: $\langle T \rangle$ ::= "**[**" | "**|**" | "**]**" | ...
- Small step semantics
  - "balancing" has to be managed by the semantics, since the grammar is now weaker

Tinycals: syntax

$$
\begin{array}{lll}
\langle S \rangle & ::= & \\
 & & \langle B \rangle \\
 & | & \text{``}\textbf{.}\text{''} \\
 & | & \text{``}\textbf{;}\text{''} \\
 & | & \text{``}\textbf{[}\text{''} \\
 & | & \text{``}\textbf{|}\text{''} \\
 & | & i_1, \ldots, i_n \text{ ``}\textbf{:}\text{''} \\
 & | & \text{`` } * \textbf{ :}\text{''} \\
 & | & \text{``}\texttt{skip}\text{''} \\
 & | & \text{``}\textbf{]}\text{''} \\
 & | & \text{``}\texttt{focus}\text{''}\ [g_1; \cdots; g_n] \\
 & | & \text{``}\texttt{done}\text{''}
\end{array}
$$

$$
\begin{array}{lll}
\langle L \rangle & ::= & \\
 & & \langle S \rangle \\
 & | & \langle S \rangle\ \langle S \rangle \\
\langle B \rangle & ::= & \\
 & & \langle T \rangle \\
 & | & \text{``}\texttt{try}\text{''}\ \langle B \rangle \\
 & | & \text{``}\texttt{repeat}\text{''}\ \langle B \rangle \\
 & | & \langle B \rangle \text{ ``}\textbf{;}\text{''} \langle B \rangle \\
 & | & \langle B \rangle \text{ ``}\textbf{;[}\text{''} \langle B \rangle \text{ ``}\textbf{|}\text{''} \ldots \text{ ``}\textbf{|}\text{''} \langle B \rangle \text{ ``}\textbf{]}\text{''} \\
\langle T \rangle & ::= & \text{``}\textit{apply}\text{''} \\
 & | & \text{``}\textit{rewrite}\text{''} \\
 & | & \ldots
\end{array}
$$

Tinycals: semantics (1/6)

type ξ $\quad$ (* *proof status* *)
type *goal*
val *apply_tac* : $\langle B \rangle \rightarrow \xi \rightarrow goal \rightarrow \xi \times goal\ \texttt{list} \times goal\ \texttt{list}$

Tinycals: semantics (2/6)

$$
\begin{array}{rcll}
\mathit{task} & = & \mathtt{int} \times (\mathtt{O}\ \mathit{goal} \mid \mathtt{C}\ \mathit{goal}) & \text{(task)} \\
\Gamma & = & \mathit{task}\ \mathtt{list} & \text{(context)} \\
\tau & = & \mathit{task}\ \mathtt{list} & \text{(``todo'' list)} \\
\kappa & = & \mathit{task}\ \mathtt{list} & \text{(dot's continuation)} \\
\mathit{tag} & = & \mathtt{B} \mid \mathtt{F} & \text{(stack level tag)} \\
\mathit{stack} & = & (\Gamma \times \tau \times \kappa \times \mathit{tag})\ \mathtt{list} & \text{(context stack)} \\
\mathit{code} & = & \langle S \rangle\ \mathtt{list} & \text{(statements)} \\
\mathit{status} & = & \mathit{code} \times \xi \times \mathit{stack} & \text{(evaluation status)}
\end{array}
$$

Tinycals: semantics (3/6)

$$\langle\langle B\rangle :: c, \xi, \langle \Gamma, \tau, \kappa, t\rangle :: S\rangle \;\longrightarrow\; \langle c, \xi_n, S'\rangle$$

where $[g_1; \cdots ; g_n] = \mathit{get\_O\_goals\_in\_tasks\_list}(\Gamma)$

$$\text{and} \begin{cases} \langle \xi_0, G_0^o, G_0^c\rangle = \langle \xi, [\,], [\,]\rangle \\ \langle \xi_{i+1}, G_{i+1}^o, G_{i+1}^c\rangle = \langle \xi_i, G_i^o, G_i^c\rangle & g_{i+1} \in G_i^c \\ \langle \xi_{i+1}, G_{i+1}^o, G_{i+1}^c\rangle = \langle \xi', (G_i^o \setminus G^c) \cup G^o, G_i^c \cup G^c\rangle & \notin \\ \quad \text{where } \langle \xi', G^o, G^c\rangle = \mathit{apply\_tac}(\langle B\rangle, \xi_i, g_{i+1}) \end{cases}$$

and $S' = \langle \Gamma', \tau', \kappa', t\rangle :: \mathit{close\_tasks}(G_n^c, S)$

and $\Gamma' = \mathit{mark\_as\_handled}(G_n^o)$

and $\tau' = \mathit{remove\_tasks}(G_n^c, \tau)$

and $\kappa' = \mathit{remove\_tasks}(G_n^c, \kappa)$

$$\langle \text{“;”} :: c, \xi, S\rangle \;\longrightarrow\; \langle c, \xi, S\rangle$$

Tinycals: semantics (4/6)

$$\langle \text{"}\mathtt{skip}\text{"} :: c, \xi, \langle \Gamma, \tau, \kappa, t\rangle :: S\rangle \;\longrightarrow\; \langle c, \xi, S'\rangle$$

where $\Gamma = [\langle j_1, \mathtt{C}\ g_1\rangle; \cdots ; \langle j_n, \mathtt{C}\ g_n\rangle]$

and $G^c = [g_1;\cdots; g_n]$

and $S' = \langle [\,], \mathit{remove\_tasks}(G^c, \tau), \mathit{remove\_tasks}(G^c, \kappa), t\rangle$
$:: \mathit{close\_tasks}(G^c, S)$

$$\langle \text{"}\mathtt{.}\text{"} :: c, \xi, \langle \Gamma, \tau, \kappa, t\rangle :: S\rangle \;\longrightarrow\; \langle c, \xi, \langle [l_1], \tau, [l_2;\cdots; l_n] \cup \kappa, t\rangle :: S\rangle$$

where $\mathit{get\_O\_tasks}(\Gamma) = [l_1;\cdots; l_n]$

$$\langle \text{"}\mathtt{.}\text{"} :: c, \xi, \langle \Gamma, \tau, l :: \kappa, t\rangle :: S\rangle \;\longrightarrow\; \langle c, \xi, \langle [l], \tau, \kappa, t\rangle :: S\rangle$$

where $\mathit{get\_O\_tasks}(\Gamma) = [\,]$

Tinycals: semantics (5/6)

$$\langle \text{``}\mathbf{[}\text{''} :: c, \xi, \langle [l_1;\cdots; l_n], \tau, \kappa, t\rangle :: S\rangle \;\longrightarrow\; \langle c, \xi, S'\rangle$$

when $\mathit{renumber\_branches}([l_1;\cdots;l_n]) = [l'_1;\cdots;l'_n]$

and $S' = \langle [l'_1], [\,], [\,], \mathtt{B}\rangle :: \langle [l'_2;\cdots;l'_n], \tau, \kappa, t\rangle :: S$

$$\langle \text{``}\mathbf{|}\text{''} :: c, \xi, \langle \Gamma, \tau, \kappa, \mathtt{B}\rangle :: \langle [l_1;\cdots; l_n], \tau', \kappa', t'\rangle :: S\rangle \;\longrightarrow\; \langle c, \xi, S'\rangle$$

where $S' = \langle [l_1], \tau \cup \mathit{get\_O\_tasks}(\Gamma) \cup \kappa, [\,], \mathtt{B}\rangle :: \langle [l_2;\cdots;l_n], \tau', \kappa'$

$$\langle i_1, \ldots, i_n \text{``}\mathbf{:}\text{''} :: c, \xi, \langle [l], \tau, [\,], \mathtt{B}\rangle :: \langle \Gamma', \tau', \kappa', t'\rangle :: S\rangle \;\longrightarrow\; \langle c, \xi, S'\rangle$$

where $\mathit{unhandled}(l)$

and $\forall j = 1\ldots n, \quad \exists l_j = \langle j, s_j\rangle, \quad l_j \in l :: \Gamma'$

and $S' = \langle [l_1;\cdots;l_n], \tau, [\,], \mathtt{B}\rangle :: \langle (l :: \Gamma') \setminus [l_1;\cdots;l_n], \tau', \kappa', t'\rangle :: S$

Tinycals: semantics (6/6)

$$\langle \text{``}\ast\textbf{:}\text{''} :: c, \xi, \langle [l], \tau, [\,], \mathtt{B}\rangle :: \langle \Gamma', \tau', \kappa', t'\rangle :: S\rangle \longrightarrow \langle c, \xi, S'\rangle$$

where $\mathit{unhandled}(l)$

and $S' = \langle l :: \Gamma', \tau, [\,], \mathtt{B}\rangle :: \langle [\,], \tau' \cup \mathit{get\_O\_tasks}(\Gamma) \cup \kappa, \kappa', t'\rangle :: S$

$$\langle \text{``}\textbf{]}\text{''} :: c, \xi, \langle \Gamma, \tau, \kappa, \mathtt{B}\rangle :: \langle \Gamma', \tau', \kappa', t'\rangle :: S\rangle \longrightarrow \langle c, \xi, S'\rangle$$

where $S' = \langle \tau \cup \mathit{get\_O\_tasks}(\Gamma) \cup \Gamma' \cup \kappa, \tau', \kappa', t'\rangle :: S$

$$\langle \text{``}\mathtt{focus}\text{''}\ [g_1; \cdots; g_n] :: c, \xi, \langle \Gamma, \tau, \kappa, t\rangle :: S\rangle \longrightarrow \langle c, \xi, S'\rangle$$

where $g_i \in \mathit{get\_O\_goals\_in\_status}(S)$

and $S' = \langle \mathit{mark\_as\_handled}([g_1; \cdots; g_n]), [\,], [\,], \mathtt{F}\rangle$

$:: \mathit{close\_tasks}(\langle \Gamma, \tau, \kappa, t\rangle :: S)$

$$\langle \text{``}\mathtt{done}\text{''} :: c, \xi, \langle [\,], [\,], [\,], \mathtt{F}\rangle :: S\rangle \longrightarrow \langle c, \xi, S\rangle$$

Demo: property_sigma.ma

demo

What about **try**, **repeat**, ...

Consider $\Gamma = [l_1; l_2]$ and the command **try** (tac1; tac2).

Think of the (unfortunate) case in which tac1 on l_1 instantiates l_2.

Then, if tac2 fails on l_1 but has success on l_2, what is the expected semantics?

- for sure **try** (tac1; tac2) should have no effect on l_1
- but the system already displayed some progress on l_1
- and skipping tac1 on l_1 may change the result of tac1 on l_2

The (right?) types for tactics

Matita 0.5 adopted a conservative type for tactics

- tac: goal $*$ status $\rightarrow$ goal list $*$ status

Matita 1.0 (will) unifies the type of tactics and tacticals

- tac: goal list $*$ status $\rightarrow$ goal list $*$ status

We then have

- focus: tactic $\rightarrow$ goal $\rightarrow$ old_tactic
- distribute : old_tactic $\rightarrow$ tactic

Gain

- **auto** on a cluster of dependent goals
- high-level management commands (postpone, regroup, clusterize)
- eases the implementation of some declarative idioms

UTF-8: input

Displaying UTF-8 is easy. What's hard is a comfortable input of UTF-8.

name	input	result
\TeX	\Rightarrow	$\Rightarrow$
	\alpha	α
Ligatures	$=>$	$\Rightarrow$
	$->$	$\rightarrow$
Alternatives	*a*	α **a**
	P	Π $\mathcal{P}$ $\mathbb{P}$
Memory	x	last alternative for x you used

Demo: utf8.ma

demo

MathML

Mathematical Markup Language (MathML) is an XML language for describing mathematical content and its presentation.

- (UTF-8) symbols
- 2-D notations
- Colors

2-D notations

?5

$a : \mathbf{N}$
$b : \mathbf{N}$
$n : \mathbf{N}$

$$(a+b)^n = \sum_{k < S\ n} \binom{n}{k} \cdot a^{(n-k)} \cdot b^k$$

OMDoc

OMDoc (Open Mathematical Documents) is a semantic markup format for mathematical documents.

OMDoc allows for mathematical expressions on three levels:

Object level formulae, written in Content MathML, OpenMath or similar

Statement level definitions, theorems, proofs, examples ...

Theory level A theory is a set of contextually related statements

History

Input

- Ambiguity support
- Tinycals
- UTF-8 support

Output

- MathML & friends
- Proof rendering
- GtkMathView
- Graphs

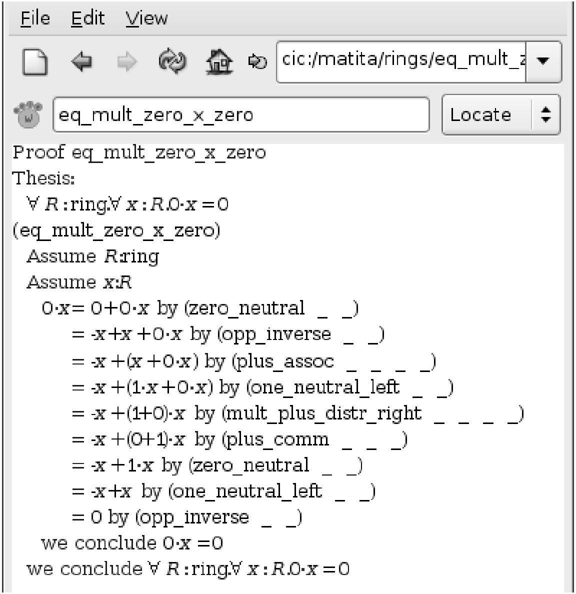
Metadata

- What's interesting about formal proofs?

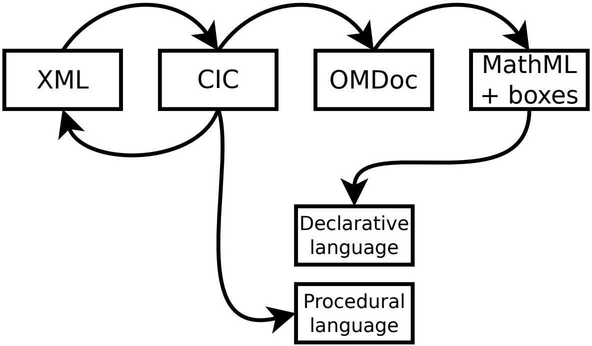
Natural language output (and input) (1/2)

Proof times_n_Sm
Thesis:
$\forall\, n:\mathrm{nat}.\forall\, m:\mathrm{nat}.n+n*m=n*S\ m$
(times_n_Sm)
Assume n:nat
Assume m:nat
we proceed by induction on n
to prove $n+n*m=n*S\ m$
Case $O \Rightarrow$
the thesis becomes $O+O*m=O*S\ m$
by (refl_eq _ _)
we conclude $O=O$
that is equivalent to $O+O*m=O*S\ m$
Case S n1:nat $\Rightarrow$
the thesis becomes S n1+S n1$*m=S$ n1$*S\ m$
by induction hypothesis we know
(H) n1+n1$*m=$n1$*S\ m$

Natural language output (and input) (2/2)



Transformations



Demo: inline.ma

demo

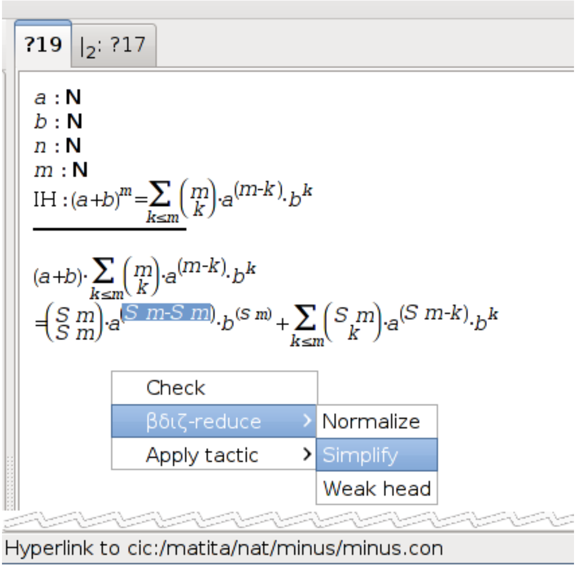
MathML widget

GtkMathView is a C++ rendering engine for MathML.
`http://helm.cs.unibo.it/mml-widget/`

Gives us, in addition to MathML rendering:

- Semantic selection
- Point and click
- Hypertext
- Alternative notations

Point and click



Demo: natural_deduction.ma

demo

Directed graphs

Some data can be displayed by means of a directed graph:

- coercions
- dependencies between scripts
- dependencies between developments

Graphviz (dot) can generate "click-able" graphs

Demo: coercions.ma

demo

Non-directed graphs

"Equivalence" classes can be displayed by means of a graph:

- unification hints

Demo: hints.ma

demo

History

Input

- Ambiguity support
- Tinycals
- UTF-8 support

Output

- MathML & friends
- Proof rendering
- GtkMathView
- Graphs

Metadata

- What's interesting about formal proofs?

Metadata (or machine understandable data)

- Last year I was hired by "mathematicians" to formalize their mathematics!
- They never asked: "Was my theorem OK?"
- But they asked me a lot of questions that Matita was (and still is) unable to answer to

Data

What can Matita do with proof terms?

- Search
- Dependencies
- ... nothing more ...

Demo: deps-search.ma

demo

What's next?

What will be dropped/kept/improved in Matita 1.0?

- Improved: tactics, tinycals and proof language (all small step)
- Improved: script file format (richer, with hyperlinks)
- Dropped: proof rendering (plugin)
- Dropped: MathML (plugin?)
- Dropped: XML (as the primary storage format)
- Kept (re-implemented): semantic selection, proof by click
- Kept: graphs

Thanks

Thanks!