Focus Windows:

A New Technique for Proof Presentation

F. Piroi, B. Buchberger

RISC-Linz
Research Institute for Symbolic Computation
A-4232 Hagenberg, Austria
{Florina.Piroi, Bruno.Buchberger}@risc.uni-linz.ac.at

1 Introduction

The idea of focus windows as a technique for proof presentation was introduced in [Buchberger 2000]. The present paper describes the implementation of the technique in the frame of the *Theorema* system [Buchberger et al. 2000]. The implementation was done by the first author under the guidance of the second author.

Proofs in mathematical publications are linear texts. In each proof step, a new formula is derived from formulae appearing in earlier portions of the text using an inference rule. Typically, in long proofs, the formulae used in a proof steps occur a couple of lines, paragraphs, or even pages distant from the place in the text at which the proof step occurs. Reference to these formulae is usually done by labels and the reader has to jump back and forth between the formulae referenced and the proof step in which they are needed. This is unpleasant and makes understanding of proofs quite difficult even if the proofs are nicely structured and well presented.

Most automated theorem provers do not put emphasis on producing proofs that are easy to read and understand. (A very telling illustration of this is provided by the collection of proofs produced for the irrationality of $\sqrt{2}$ by 15 different current prover in [Weijdeck 2001].) However, even those that provide tools for studying proofs have the problem described above. This is even true for natural deduction provers as, for example, the Omega system [Omega], that display the entire proof tree in tree form and not in linearized presentation: Still, when studying one particular proof step, one may have to jump to various formulae in the tree for being able to check or understand the validity of the particular step.

From the outset, in *Theorema* we tried to emphasize attractive proof presentation. *Theorema* proofs are designed to resemble proofs generated by humans, i.e.they contain formulae and explanatory text in English. In addition, *Theorema* provides various tools for helping the reader to browse the proofs: links to labeled formulae are realized as hyperlinks that display the formula referenced in a small auxiliary window; nested brackets at the right-hand margin make it possible to contract entire sub-proofs to just one line; various color codes distinguish the (temporary) proof goals from formulae in the (temporary) knowledge base; etc. Still, reading and understanding linear proofs is difficult even for proofs generated by the typical *Theorema* provers.

Focus windows provide a means to overcome this difficulty. The technique could be implemented for most every proving system, even for systems that do not actually generate proofs automatically but restrict automation to checking proofs generated by humans (proof checkers like HOL [Hol], MIZAR [Mizar]). The only prerequisite for applying the focus windows technique is that the results of the provers in the system must be formal proof objects that contain, for each proof step, the information on which formulae are used and which formulae are produced in the given step. The idea of the focus windows technique is simple but helpful: Starting from the proof objects, one analyzes, in each proof step, which formulae are used and which ones are produced in the given step. One then composes a window containing exactly these formulae for each proof step. The windows also contain a button for moving to the next window in the proof. In proof situations that branch to two or more proof situations the subsequent windows are displayed in contracted form and the user can decide which one to open next. In addition, each focus window contains a small simplified image of the entire proof tree so that the user can also jump to a randomly chosen proof situation in the proof tree if he wants to interrupt the sequence of proof steps suggested by the system.

In more detail, a proof situation in a proof generated by one of the *Theorema* provers (and also the provers in most other proving system) consists of

- the current proof goal and
- the current knowledge base (definitions, axioms, known theorems, and temporary assumptions).

In one proof step, using only a few formulae in the knowledge base, which we call here the "relevant formulae", by applying one of the inference techniques of the prover a new goal or new formulae in the knowledge base or both are generated. In the corresponding focus window we display

- the current proof goal and
- the relevant formulae

and then, upon clicking, we display in addition the new goal and / or the new formulae in the knowledge base. When clicking the next button, we then display the focus window for the next proof situation.

Another way of describing the focus window technique is to think about a long proof written on a sequence of blackboards. Now, for reading and understanding the proof, you equip the reader with a magic glass (a "focus glass") that, in each proof step, concentrates all the relevant formulae on one blackboard and erases all the other formulae so that the reader, in each proof step, has all the relevant information in front of his eyes and is not distracted by any irrelevant material.

Note that this technique is applicable for both general and special inference rules: For example, if polynomial simplification or even advanced techniques like cylindric algebraic decomposition or symbolic summation are black-box inference rules in the prover applied to a particular proof problem then, for each proof step, the corresponding focus window will only show the proof goal and the relevant formulae before applying the special inference rule and the formulae produced by this inference rule and will not show any internal details of the execution of the inference rule.

The focus windows method is most easily understood by looking to the examples in the appendix. Appendix 2 shows the linear proof text generated by the current version of the *Theorema* system in an automatic post-processing step applied to the abstract proof object automatically generated by one of the *Theorema* provers. Appendix 3, in comparison, shows the sequence of focus windows generated, again in an automatic post-processing step applied to the abstract proof object.

In Section 2 we will give some comments on the focus windows appearing in the example in Appendix 3 in comparison to the proof shown in Appendix 2. In Section 3, we will then give some comments on the implementation of the focus windows technique in *Theorema*.

2 Using Focus Windows

The example used to explain the focus windows technique is part of a *Theorema* case study that covers the theory of equivalence relations, equivalence classes, partitions and induced relations, [Windsteiger 2001b]. All notions are defined in terms of set theory. Appendix 2 presents the output of a **Prove** call of *Theorema* that generates the proof of a simple theorem on equivalences and partitions automatically.

We do not go here into the details of the *Theorema* language. The formulae should be self–explanatory. All the knowledge the user has to type in order to obtain a successful proof is presented in Appendix 1.

A typical call for starting a *Theorema* prover to work on a proof problem looks like this:

```
Prove [Lemma["Lemma"],  using → KnowledgeBase,
  by → SomeProver,  ProverOptions → {options for the prover},
  showBy → SomeDisplayer ]
```

The user of *Theorema* can control both the work of SomeProver by setting the ProverOptions and also the way the proof is presented by setting the showBy options. By default, *Theorema* will present the proof in a new *Mathematica* notebook as a linear proof text. By setting *showBy→FocusWindow* the focus window display method will be invoked, which is the topic of this paper. (For a complete description of the options of the Prove command and other details about *Theorema* see [Windsteiger 2001a], [Vasaru 2000]).

For example, the following command generates the linear text proof presented in Appendix 2

```
Prove[Lemma["Lemma"],  using →
    〈Definition["factor set"], Definition["is all non empty"], Definition["class"],
      Definition["is reflexive"], Definition["is relation"],  Definition["is subset set"]〉,
  by → SetTheoryPCSProver (∗ , other options ∗)],
```

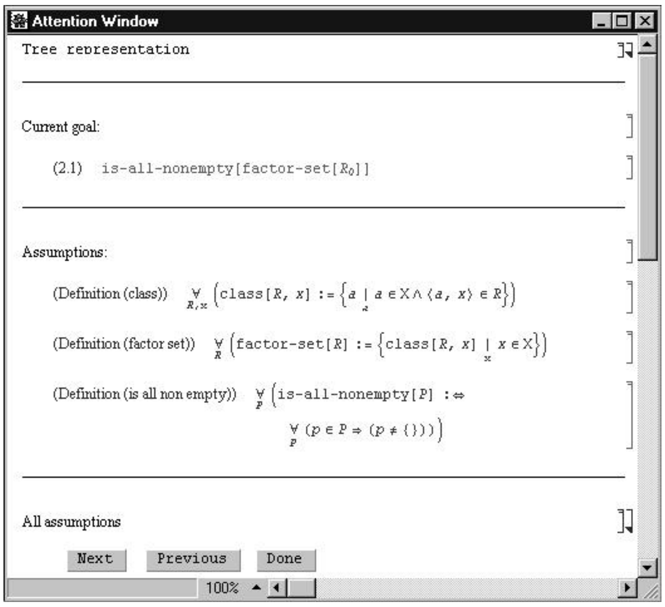
and the following command generates the focus window presentation of the same proof:

```
Prove[Lemma["Lemma"],  using →
    〈Definition["factor set"], Definition["is all non empty"], Definition["class"],
      Definition["is reflexive"], Definition["is relation"],  Definition["is subset set"]〉,
  by → SetTheoryPCSProver,  showBy → FocusWindow (∗ , other options ∗)],
```

As mentioned earlier in this paper, the focus window method presents proofs in a step–wise manner. Each step of the proof will be shown to the user in two phases: the attention phase and the transformation phase with a corresponding Attention Window and a Transformation Window. Each of these windows has

- a "goal area" in which the current goals are shown,
- an "assumptions area" in which the "relevant" assumptions are shown,
- a "proof tree area" in which the entire proof tree is displayed in a schematic, simplified form,
- an area that presents all the assumptions that are available (the "all assumptions area"),
- and a "navigation area" that helps the user navigate in the proof by clicking on various buttons.

As a concrete example, the goal area of the Attention Window 5 (Figure 1) shows us the current goal formula (2.1) and the assumption area shows the definitions of the functions 'class' and 'factor–set' and the predicate 'is–all–nonempty'. The area containing the schematic representation of the proof tree and the area containing all the assumptions that are currently available are shown in closed cells. If the user is interested to see the contents of these cells it has to double–click on the respective cell brackets. (The organization of notebooks using cells is a standard *Mathematica* feature, see [Wolfram 96]). Note that, following the basic philosophy of the focus windows technique, the user will normally *not* want to see all assumptions but only the ones that are *relevant* for the current proof step, which are exactly the ones shown in the assumptions area.



Tree representation

Current goal:

(2.1) is-all-nonempty[factor-set[R_0]]

Assumptions:

(Definition (class)) $\underset{R,x}{\forall}\left(\text{class}[R, x] := \left\{a \underset{a}{|} a \in X \wedge \langle a, x\rangle \in R\right\}\right)$

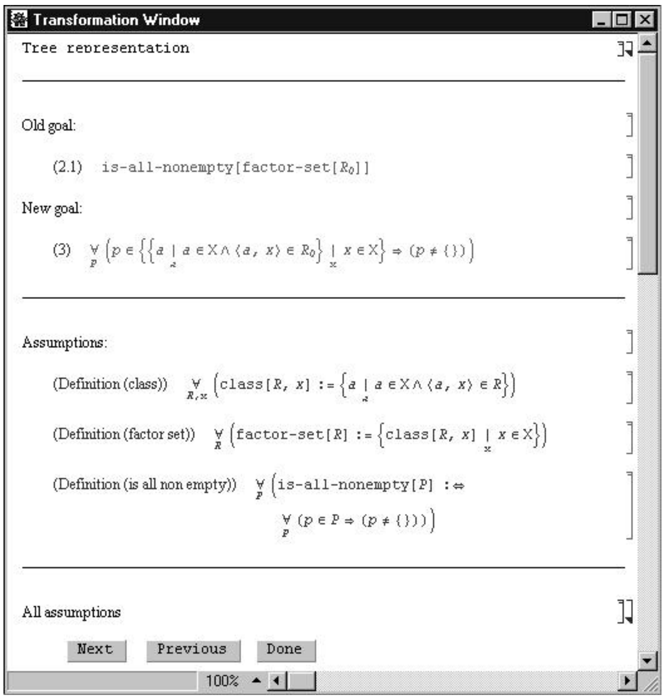
(Definition (factor set)) $\underset{R}{\forall}\left(\text{factor-set}[R] := \left\{\text{class}[R, x] \underset{x}{|} x \in X\right\}\right)$

(Definition (is all non empty)) $\underset{P}{\forall}\left(\text{is-all-nonempty}[P] :\Leftrightarrow \underset{p}{\forall}(p \in P \Rightarrow (p \neq \{\}))\right)$

All assumptions

Figure 1

A click on the 'Next' button will bring up the Transformation Window 5 (Figure 2). The inference step applied in the current proof step rewrote the goal using the three definitions in the assumptions area - and no other formulae of the current knowledge base! The result of rewriting the goal is now displayed as formula (3) under the heading 'New goal:'. Formula (2.1) is now an 'Old goal'. No new assumptions were inferred in this step. Therefore, in this example, the assumptions area does not contain any new formulae. (In contrast, in the Attention and Transformation Window 2 – Appendix 3 –, new assumptions are generated, which are displayed under the heading 'New assumptions:'.)



Tree representation

Old goal:

(2.1) is-all-nonempty[factor-set[R_0]]

New goal:

(3) $\underset{p}{\forall}\left(p \in \left\{\left\{a \underset{a}{|} a \in X \wedge \langle a, x\rangle \in R_0\right\} \underset{x}{|} x \in X\right\} \Rightarrow (p \neq \{\})\right)$

Assumptions:

(Definition (class)) $\underset{R,x}{\forall}\left(\text{class}[R, x] := \left\{a \underset{a}{|} a \in X \wedge \langle a, x\rangle \in R\right\}\right)$

(Definition (factor set)) $\underset{R}{\forall}\left(\text{factor-set}[R] := \left\{\text{class}[R, x] \underset{x}{|} x \in X\right\}\right)$

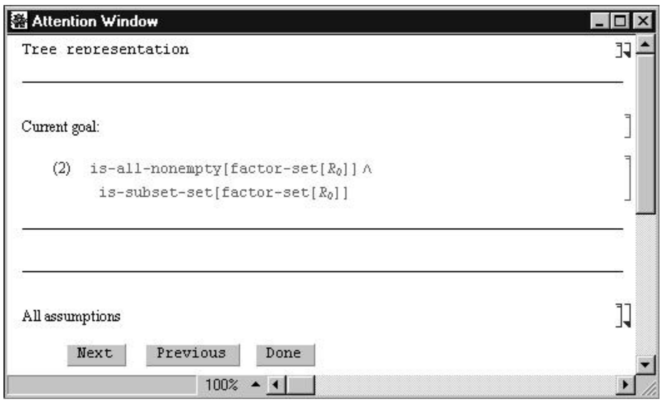
(Definition (is all non empty)) $\underset{P}{\forall}\left(\text{is-all-nonempty}[P] :\Leftrightarrow \underset{p}{\forall}(p \in P \Rightarrow (p \neq \{\}))\right)$

All assumptions

Figure 2

Pressing the 'Previous' button now will take the user back to the Attention Window 5. Clicking 'Previous' again, the user will see the Transformation Window 4. One may go back and forth as many times as it needs to understand the proof completely. (Here, "understanding" means "verifying the correctness" of each proof step and "verifying the completeness of the sequence of proof steps". In this paper, we cannot go into the discussion of various notions of "understanding mathematical proofs".)

Now let us look closer at the Attention Window 4 (Figure 3). The current goal – formula (2) – is a conjunction of two statements, each of them needs to be proved. Therefore, the proof will split into two branches. The Transformation Window that comes up after clicking 'Next' (Figure 4) contains now two closed cells, one for each sub–proof. Each of the cells contains its goal area, assumption area, navigation button area and all–assumption area. The user can continue on the branch it wishes to see by clicking on the 'Next' button of the corresponding branch. There is, however, only one proof tree area in this window.



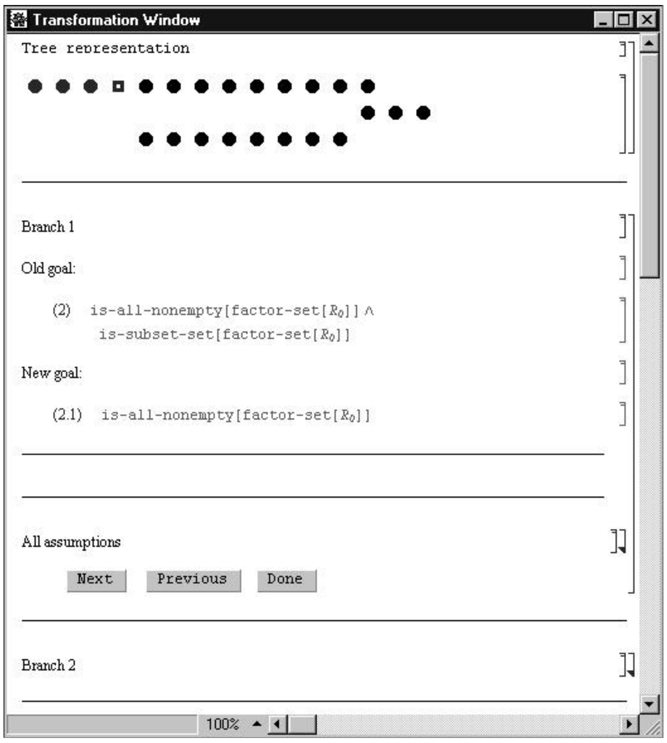
Tree representation

Current goal:

(2) is-all-nonempty[factor-set[R_0]] ∧
is-subset-set[factor-set[R_0]]

All assumptions

Figure 3



Tree representation

Branch 1

Old goal:

(2) is-all-nonempty[factor-set[R_0]] ∧
is-subset-set[factor-set[R_0]]

New goal:

(2.1) is-all-nonempty[factor-set[R_0]]

All assumptions

Branch 2

Figure 4 – the cell corresponding to the first branch is opened.

The simplified proof representation in the proof tree area is not only a graphical representation but it also has some functionality. The nodes of the simplified tree representation are in one–to–one correspondence with the nodes of the proof object. The node corresponding to the proof step that is currently seen in either the Attention or the Transformation Window is high–lighted. Also, clicking any of these nodes will cause the corresponding focus window to be displayed, allowing the user to read the proof in the order it prefers. (In contrast, using the buttons in the navigation, the proof can be read "forward" and "backward" in the sequence suggested by the prover.)

3 Implementation of Focus Windows

The proof presentation technique explained in the section above should not be difficult to implement in any existing automated prover. The main pre–requisite is that the data structure for the proof object, for each proof step, contains sufficient information for extracting the formulae used and inferred in the particular step. The Focus Windows technique can then be described, roughly, by the following pseudo-algorithm:

```
Step 1: phase = initial;
        current_position = root_position;
        ShowWindow[phase, current_position, proof_object];
Step 2: while user_action is not done
          if phase is initial or transformation
              then phase = attention;
              else phase = transformation;
          end if;
          current_position =
              DetermineCurrentPosition[current_position,
                                       user_action, proof_object];
          ShowWindow[phase, current_position, proof_object];
        end while;
Step 3: stop.
```

Note that the value of 'proof_object' does not change during the execution of the algorithm: The proof object is the essential input parameter of the algorithm. The variables used in this pseudo–algorithm are 'phase', 'current_position' and 'user_action'. The variable 'phase' may have the values '*initial*', '*attention*', or '*transformation*'. The 'current_position' points to the current proof step in the proof object is. The values 'current_position' may take depend on the exact data structure of the proof objects. We assume here that the proof object is in a tree–like form but the algorithm may be easily adapted to any other data structure. The 'user_action' variable is needed to take in the user actions (button clicks, keys on the keyboard, words typed in at the prompter, etc.). The constants appearing in the algorithm are in italics.

The core of the algorithm is the ShowWindow function. At the very beginning, it is called with the initializing parameters *initial* and *root_position*. In the *initial* phase, we generate the initial focus window that contains the goal that has to be proved and the available knowledge (axioms, definitions, lemmata, temporary assumptions, etc.). When the user takes an action three main steps are executed:

- Depending on the actual phase, the type of the next phase is set.
- Depending on the 'current_position' and the 'user_action', the algorithm determines the position in the proof object of the next proof step on which we want to focus on (DetermineCurrentPosition).
- The focus window corresponding to the new 'current_position' and 'phase' is shown.

We implemented the focus window method in *Mathematica* [Wolfram 96], which is also the language we chose for the implementation of *Theorema*. In fact, the implementation was quite easy because of two reasons:

- From the outset, the data structure of *Theorema* proof objects was carefully designed in order to give easy access to the formulae relevant in each proof step. Also, the data structure for *Theorema* proof objects leaves some slots open for adding additional information which is relevant for certain prove methods or certain proof presentation methods.
- The front end of *Mathematica* provides convenient programming tools for active objects that, basically, allow to apply the usual *Mathematica* programming style also for programming man–machine interfaces. We use this facility for entering the information in the action buttons and the schematic proof tree representation into the ShowWindow function. We give some more details about this below.

The user actions are taken in via the buttons 'Next', 'Previous' and 'Done' in the navigation area and the schematic proof tree presentation whose nodes are, in fact, also realized as buttons. The schematic proof tree representation is a static object in the sense that the data attached to its node buttons do not change during the presentation of the proof by the focus window viewer. In contrast, the buttons 'Next' and 'Previous' are dynamic objects, whose information is used in the following way:

- Suppose that the focus window is presenting the Attention Window of some node *n* of the proof tree. Then the data attached to the 'Previous' button is a link to the parent node of *n*. The data attached to the 'Next' button is a link to the node *n* because when pressing it we want to bring up the Transformation Window of the same node *n*.
- Suppose that the focus window is presenting the Transformation Window of some node *n* of the proof tree. Note that such a window may have several branches. Then the data attached to the 'Previous' button in each of the

branches is a link to the node *n* because when pressing it we want to bring up the Attention Window of the same node *n*. The data attached to the 'Next' button in each of the branches is a link to the corresponding child node of *n*.

4 Conclusion

We presented a new method for presenting mathematical proofs that could be applied, in principle, to both proofs generated by humans and proofs generated by algorithms. The essence of the method is that we present, in each proof step, exactly the formulae that are *relevant* for the particular proof step and we put these formulae into our *focus*. If proofs are printed on paper or written on other passive media like blackboards, this operation of *focusing* cannot be executed automatically (but is done "in the head of the person who studies a proof"). In contrast, in the context of automated theorem proving, when proofs are naturally available as processable data objects (the "proof objects") this focusing operation can be described by an algorithm and can be made available for the users of automated theorem proving systems. We described an implementation of this new tool in the frame of the *Theorema* project.

Note that the focus windows tool starts from proof objects and just implements a particular way of presenting proof objects. The focus window tool is *not* a prove method!

Note also that the focus windows tool is applicable for any type of prover including special provers that have powerful special proof rules that apply only for certain mathematical theories. For example, simplification modulo an AC-theory may be a black–box proof rule in a predicate logic natural deduction prover. Thus, the focus windows technique does not assert that each of the proof steps should be "easily" verifiable but, rather, it just gives a method to keep track of the relevant information used in each proof step the particular prover generates.

When comparing the linear proof presentation and the focus window proof presentation of one and the same proof one may make the following observations:

- In short proofs, the focus windows presentation may generate presentation overhead that will distract the reader rather than help him.
- In proofs that are more than one or two pages long, the focus window presentation may increase the possibility of verifying proofs drastically.
- Linear presentations are helpful for obtaining a quick overview on the overall flow of the proof whereas focus window presentations support the process of verifying proofs.

- Most probably, browsing a proof in linear representation and, then, verifying the details of the proof in focus window presentation is the most reasonable and efficient way of studying proofs.

The proof given as an example in the appendices is just borderline: The positive effect of the focus window presentation on verifiability is noticeable but not yet drastic. However, by space limitation, it is not possible to present longer proofs in this paper. Thus, we encourage the reader to experiment with the *Theorema* system and its linear and focus windows proof presentation tools.

After having implemented the focus window technique in *Theorema*, we also made another, interesting and unexpected, experience: The tool can of course be applied also to wrong proofs. In particular it can be used to check the proofs generated by theorem provers that are under construction and not yet fully tested. Here we noticed that checking the proofs by the focus window technique makes it much easier to detect errors in the provers. Thus, the focus window tool may also be a useful research instrument for people working in the design and implementation of automated theorem provers.

In the next version of our focus window tool,we intend to add a couple of improvements:

- The size of the font of the formulae in a focus window will be adjusted in such a way that, always ,the entire focus window will fit into the screen for facilitating having the overview on the current proof situation.
- A standard text will appear in each focus window that indicates the proof technique used for going from the formulae in the attention window to the formulae in the transformation window. This text will be a hyperlink that leads the reader,if it wants,to a detailed explanation of the proof technique.
- The focus window will be an option that can be clicked when studying a proof in linear notation.In other words,when clicking into a proof goal in a linear proof text,the corresponding focus window (attention window and then transformation window) should appear that contains exactly the formulae relevant in the current proof situation. Maybe, this way of using focus windows will be the most attractive combination of tranditional linear presentation and the new focus window technique.

References

[Buchberger 2000] *Focus Windows Presentation: A New Approach to Presenting Mathematical Proofs (in Automated Theorem Proving Systems). Theorema* Technical Report, 2000-01-30, RISC, http://www.risc.uni-linz.ac.at/people/buchberg/downloads.html

[Buchberger et al. 2000] B. Buchberger, C. Dupré, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, W. Windsteiger. *The Theorema Project: A Progress Report.* In: Symbolic Computation and Automated Reasoning (Proceedings of CALCULEMUS 2000, Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, August 6-7, 2000, St. Andrews, Scotland, M. Kerber and M. Kohlhase eds.), A.K. Peters, Natick, Massachusetts, pp. 98-113. ISBN 1-56881-145-4.

[Hol] *The HOL System.* Developed at the University of Cambridge, directed by R. Milner. http://www.cl.cam.ac.uk/Research/HVG/HOL/.

[Mizar] *Mizar System.* Developed at the University of Warsaw, directed by A. Trybulec. http://mizar.uwb.edu.pl/system/.

[Omega] *Omega System.* Developed at the University of Saarbrücken, directed by J. Siekmann. http://www.ags.uni-sb.de/~omega/intro.html.

[Vasaru 2000] D. Vasaru–Dupré, *Automated Theorem Proving by Integrating Proving, Solving and Computing*. RISC Institute, May 2000, RISC report 00–19. PhD Thesis.

[Windsteiger 2001a] W.Windsteiger, *A Set Theory Prover in Theorema: Implementation and Practical Applications,* RISC Institute, May 2001, RISC report 01–03. PhD Thesis.

[Windsteiger 2001b] W. Windsteiger, *Exploring Equivalence Relations and Partitions using the Theorema Set Theory Prover*, Technical Report, RISC Institute, Mai 2001.

[Wolfram 96] S.Wolfram. *The Mathematica Book*, Wolfram Media and Cambridge University Press, 1996.

Appendices

Appendix 1

Loading up the *Theorema* system is done by evaluating the following command:

```
Needs["Theorema`"];
```

The notion of "relation" is introduced as a subset of a binary cartesian product (X being a constant):

```
Definition["is relation", any[R], is–relation[R] :⇔ R ⊆ X×X]
Definition["is reflexive", any[R],  is–reflexive[R] :⇔ ∀_{x∈X} ⟨x, x⟩ ∈ R]
```

Other definitions needed:

```
Definition["class", any[x, R], class[R, x] := {a | ⟨a, x⟩ ∈ R}]
Definition["factor set", any[R], factor–set[R] := {class[R, x] |_x}]
Definition["is subset set", any[P], is–subset–set[P] :⇔ ∀_{p∈P} p ⊆ X]
Definition["is all non empty", any[P], is–all–nonempty[P] :⇔ (∀_{p∈P} p ≠ {})]
```

The lemma we want to prove:

```
Lemma["Lemma", any[R],  (is–relation[R] ∧ is–reflexive[R]) ⇒
    (is–all–nonempty[factor–set[R]] ∧ is–subset–set[factor–set[R]])]
```

The proof presented in Appendix 2 is obtained by evaluating the following command:

```
Prove[Lemma["Lemma"],  using → ⟨Definition["factor set"],
        Definition["class"], Definition["is relation"], Definition["is reflexive"],
        Definition["is subset set"], Definition["is all non empty"]⟩,
   by → SetTheoryPCSProver, showBy → FocusWindow,
   transformBy → ProofSimplifier,  TransformerOptions → {branches → Proved, steps → Useful},
   ProverOptions → {GRWTarget → {"goal", "kb"}, UseCyclicRules → True,
        RWExistentialGoal → True, DisableProver → {STC, PND}},  SearchDepth → 50]
```

Appendix 2

Prove:

(Lemma) $\underset{R}{\forall}$ (is–relation$[R] \wedge$ is–reflexive$[R] \Rightarrow$,

is–all–nonempty[factor–set$[R]] \wedge$ is–subset–set[factor–set$[R]]$)

under the assumptions:

(Definition (factor set)) $\underset{R}{\forall}\left(\text{factor–set}[R] := \left\{\underset{x}{\text{class}[R, x] \mid x \in \mathrm{X}}\right\}\right)$,

(Definition (class)) $\underset{R,x}{\forall}\left(\text{class}[R, x] := \left\{\underset{a}{a \mid a \in \mathrm{X} \wedge \langle a, x\rangle \in R}\right\}\right)$,

(Definition (is relation)) $\underset{R}{\forall}(\text{is–relation}[R] :\Leftrightarrow R \subseteq \mathrm{X} \times \mathrm{X})$,

(Definition (is reflexive)) $\underset{R}{\forall}\left(\text{is–reflexive}[R] :\Leftrightarrow \underset{x}{\forall}(x \in \mathrm{X} \Rightarrow \langle x, x\rangle \in R)\right)$,

(Definition (is subset set)) $\underset{P}{\forall}\left(\text{is–subset–set}[P] :\Leftrightarrow \underset{p}{\forall}(p \in P \Rightarrow p \subseteq \mathrm{X})\right)$,

(Definition (is all non empty)) $\underset{P}{\forall}\left(\text{is–all–nonempty}[P] :\Leftrightarrow \underset{p}{\forall}(p \in P \Rightarrow (p \neq \{\}))\right)$.

We assume

(1) is–relation$[R_0] \wedge$ is–reflexive$[R_0]$,

and show

(2) is–all–nonempty[factor–set$[R_0]] \wedge$ is–subset–set[factor–set$[R_0]]$.

We prove the individual conjunctive parts of (2):

Proof of (2.1) is–all–nonempty[factor–set$[R_0]$]:

Formula (2.1), using (Definition (factor set)), is implied by:

is–all–nonempty$\left[\left\{\underset{x}{\text{class}[R_0, x] \mid x \in \mathrm{X}}\right\}\right]$,

which, using (Definition (class)), is implied by:

is–all–nonempty$\left[\left\{\underset{x}{\left\{\underset{a}{a \mid a \in \mathrm{X} \wedge \langle a, x\rangle \in R_0}\right\} \mid x \in \mathrm{X}}\right\}\right]$,

which, using (Definition (is all non empty)), is implied by:

(3) $\underset{p}{\forall}\left(p \in \left\{\underset{x}{\left\{\underset{a}{a \mid a \in \mathrm{X} \wedge \langle a, x\rangle \in R_0}\right\} \mid x \in \mathrm{X}}\right\} \Rightarrow (p \neq \{\})\right)$.

We assume

(4) $p_0 \in \left\{\underset{x}{\left\{\underset{a}{a \mid a \in \mathrm{X} \wedge \langle a, x\rangle \in R_0}\right\} \mid x \in \mathrm{X}}\right\}$,

and show

(5) $p_0 \neq \{\}$.

From what we already know follows:

From (4) we know by definition of $\left\{\underset{x}{T_x \mid P}\right\}$ that we can choose an appropriate value such that

(6) $a1_0 \in \mathrm{X}$,

(7) $p_0 = \left\{\underset{a}{a \mid a \in \mathrm{X} \wedge \langle a, a1_0\rangle \in R_0}\right\}$.

Formula (5) means that we have to show that

(13) $\underset{p1}{\exists}\,(p1 \in p_0)$.

Formula (13), using (7), is implied by:

(14) $\underset{p1}{\exists}\left(p1 \in \left\{\underset{a}{a \mid} a \in \mathrm{X} \wedge \langle a, a1_0\rangle \in R_0\right\}\right)$.

In order to prove (14) we have to show:

(15) $\underset{p1}{\exists}\,(p1 \in \mathrm{X} \wedge \langle p1, a1_0\rangle \in R_0)$.

Now, let $p1 := a1_0$. Thus, for proving (15) it is sufficient to prove:

(16) $a1_0 \in \mathrm{X} \wedge \langle a1_0, a1_0\rangle \in R_0$.

We prove the individual conjunctive parts of (16):

Proof of (16.1) $a1_0 \in \mathrm{X}$:

Formula (16.1) is true because it is identical to (6).

Proof of (16.2) $\langle a1_0, a1_0\rangle \in R_0$:

Formula (1.2), by (Definition (is reflexive)), implies:

(26) $\underset{x}{\forall}\,(x \in \mathrm{X} \Rightarrow \langle x, x\rangle \in R_0)$.

Formula (16.2), using (26), is implied by:

(27) $a1_0 \in \mathrm{X}$.

Formula (27) is true because it is identical to (6).

Proof of (2.2) is–subset–set[factor–set[R_0]]:

Formula (2.2), using (Definition (factor set)), is implied by:

is–subset–set$\left[\left\{\underset{x}{\mathrm{class}[R_0, x] \mid} x \in \mathrm{X}\right\}\right]$,

which, using (Definition (class)), is implied by:

is–subset–set$\left[\left\{\underset{x}{\left\{\underset{a}{a \mid} a \in \mathrm{X} \wedge \langle a, x\rangle \in R_0\right\} \mid} x \in \mathrm{X}\right\}\right]$,

which, using (Definition (is subset set)), is implied by:

(28) $\underset{p}{\forall}\left(p \in \left\{\underset{x}{\left\{\underset{a}{a \mid} a \in \mathrm{X} \wedge \langle a, x\rangle \in R_0\right\} \mid} x \in \mathrm{X}\right\} \Rightarrow p \subseteq \mathrm{X}\right)$.

We assume

(29) $p_1 \in \left\{\underset{x}{\left\{\underset{a}{a \mid} a \in \mathrm{X} \wedge \langle a, x\rangle \in R_0\right\} \mid} x \in \mathrm{X}\right\}$,

and show

(30) $p_1 \subseteq \mathrm{X}$.

From what we already know follows:

From (29) we know by definition of $\left\{\underset{x}{T_x \mid} P\right\}$ that we can choose an appropriate value such that

(31) $a8_0 \in \mathrm{X}$,

(32) $p_1 = \left\{\underset{a}{a \mid} a \in \mathrm{X} \wedge \langle a, a8_0\rangle \in R_0\right\}$.

For proving (30) we choose

(38) $p2_0 \in p_1$,

and show:

(39) $p2_0 \in \mathrm{X}$.

Formula (38), by (32), implies:

(42) $p2_0 \in \left\{a \underset{a}{\mid} a \in \mathrm{X} \wedge \langle a, a8_0 \rangle \in R_0\right\}$.

From what we already know follows:

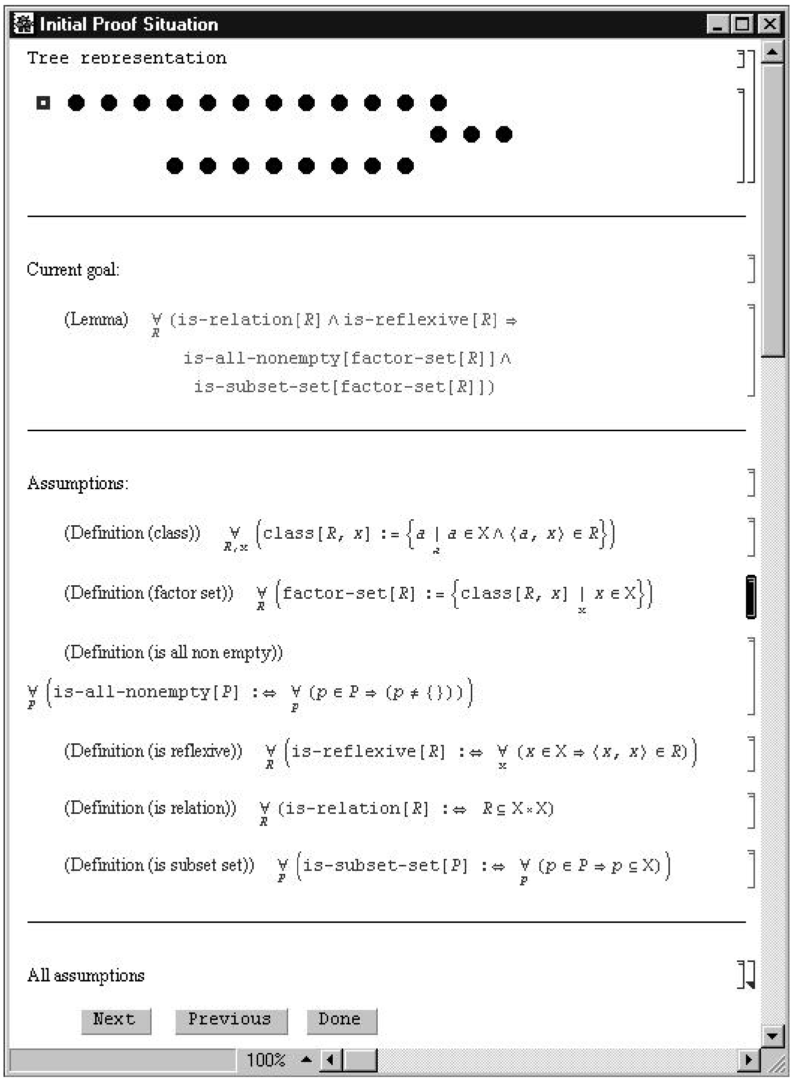
From (42) we can infer

(43) $p2_0 \in \mathrm{X} \bigwedge \langle p2_0, a8_0 \rangle \in R_0$.

Formula (39) is true because it is identical to (43.1).

Appendix 3

Appendix three presents the sequence of the focus windows, as they appear in the order proposed by the prover that generated the proof:



Current goal:

(Lemma) $\underset{R}{\forall}$ (is-relation[R] ∧ is-reflexive[R] ⇒ is-all-nonempty[factor-set[R]] ∧ is-subset-set[factor-set[R]])

Assumptions:

(Definition (class)) $\underset{R,x}{\forall}\left(\text{class}[R, x] := \left\{a \underset{a}{\mid} a \in X \land \langle a, x\rangle \in R\right\}\right)$

(Definition (factor set)) $\underset{R}{\forall}\left(\text{factor-set}[R] := \left\{\text{class}[R, x] \underset{x}{\mid} x \in X\right\}\right)$

(Definition (is all non empty))
$\underset{P}{\forall}\left(\text{is-all-nonempty}[P] :\Leftrightarrow \underset{p}{\forall}(p \in P \Rightarrow (p \neq \{\}))\right)$

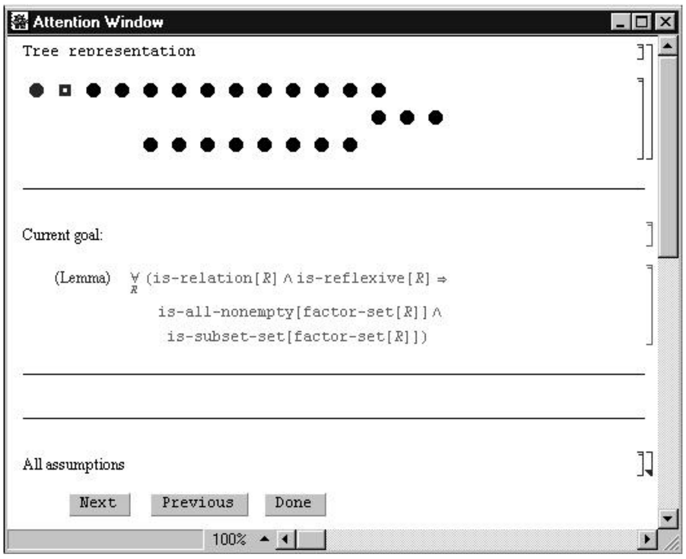
(Definition (is reflexive)) $\underset{R}{\forall}\left(\text{is-reflexive}[R] :\Leftrightarrow \underset{x}{\forall}(x \in X \Rightarrow \langle x, x\rangle \in R)\right)$

(Definition (is relation)) $\underset{R}{\forall}(\text{is-relation}[R] :\Leftrightarrow R \subseteq X \times X)$

(Definition (is subset set)) $\underset{P}{\forall}\left(\text{is-subset-set}[P] :\Leftrightarrow \underset{p}{\forall}(p \in P \Rightarrow p \subseteq X)\right)$

All assumptions

Window 1: Initial Proof Situation



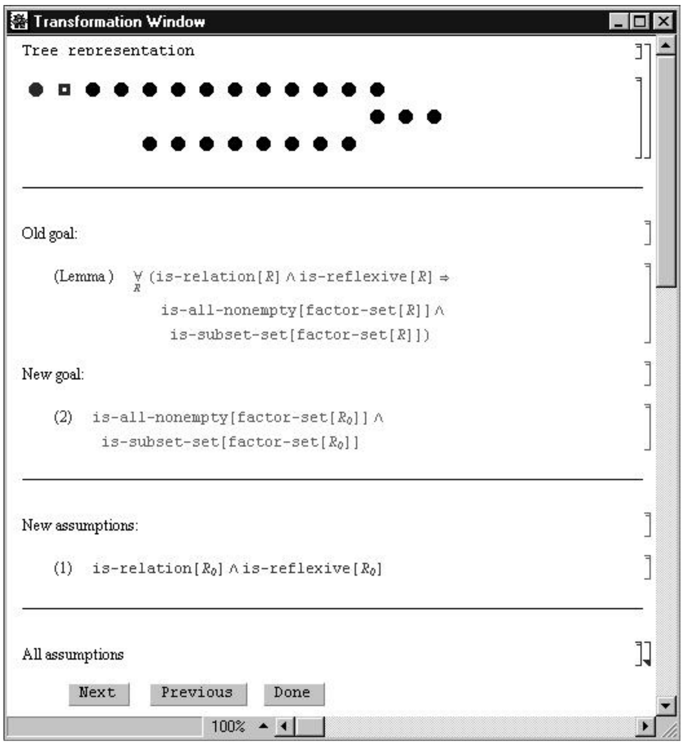
Tree representation

Current goal:

(Lemma) $\forall_{R}$ (is-relation[R] $\wedge$ is-reflexive[R] $\Rightarrow$
is-all-nonempty[factor-set[R]] $\wedge$
is-subset-set[factor-set[R]])

All assumptions

Attention Window 2



Tree representation

Old goal:

(Lemma) $\forall_{R}$ (is-relation[R] $\wedge$ is-reflexive[R] $\Rightarrow$
is-all-nonempty[factor-set[R]] $\wedge$
is-subset-set[factor-set[R]])

New goal:

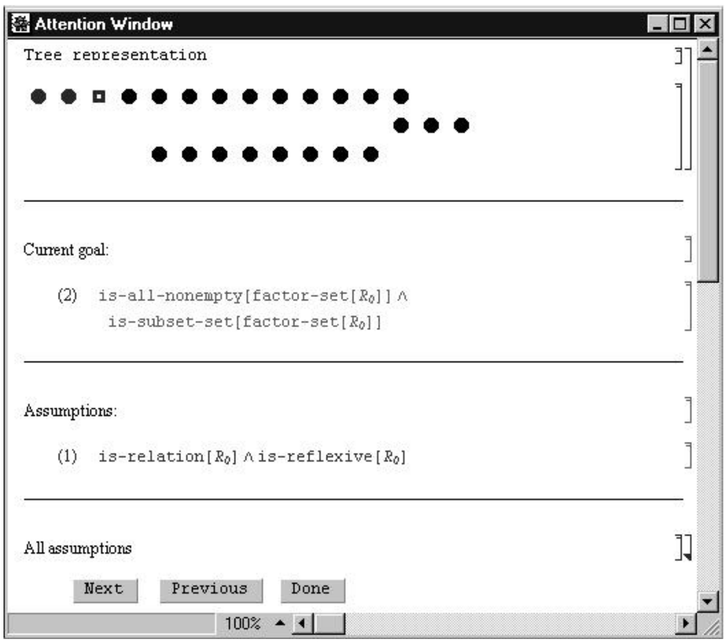
(2) is-all-nonempty[factor-set[R_0]] $\wedge$
is-subset-set[factor-set[R_0]]

New assumptions:

(1) is-relation[R_0] $\wedge$ is-reflexive[R_0]

All assumptions

Transformation Window 2



Tree representation

Current goal:

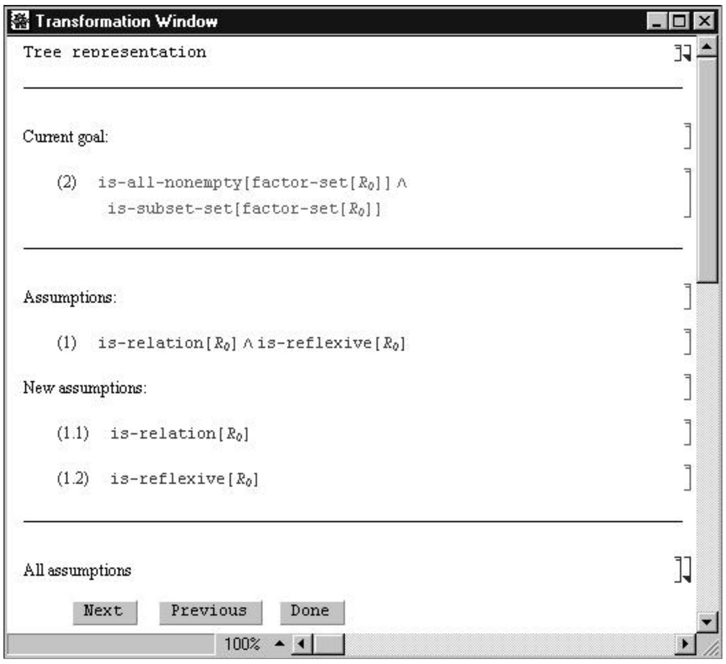
(2) is-all-nonempty[factor-set[R_0]] ∧
is-subset-set[factor-set[R_0]]

Assumptions:

(1) is-relation[R_0] ∧ is-reflexive[R_0]

All assumptions

Attention Window 3



Tree representation

Current goal:

(2) is-all-nonempty[factor-set[R_0]] ∧
is-subset-set[factor-set[R_0]]

Assumptions:

(1) is-relation[R_0] ∧ is-reflexive[R_0]

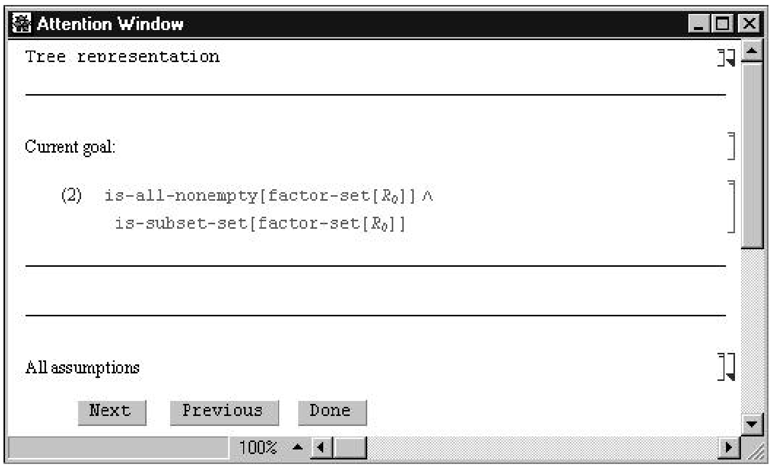
New assumptions:

(1.1) is-relation[R_0]

(1.2) is-reflexive[R_0]

All assumptions

Transformation Window 3



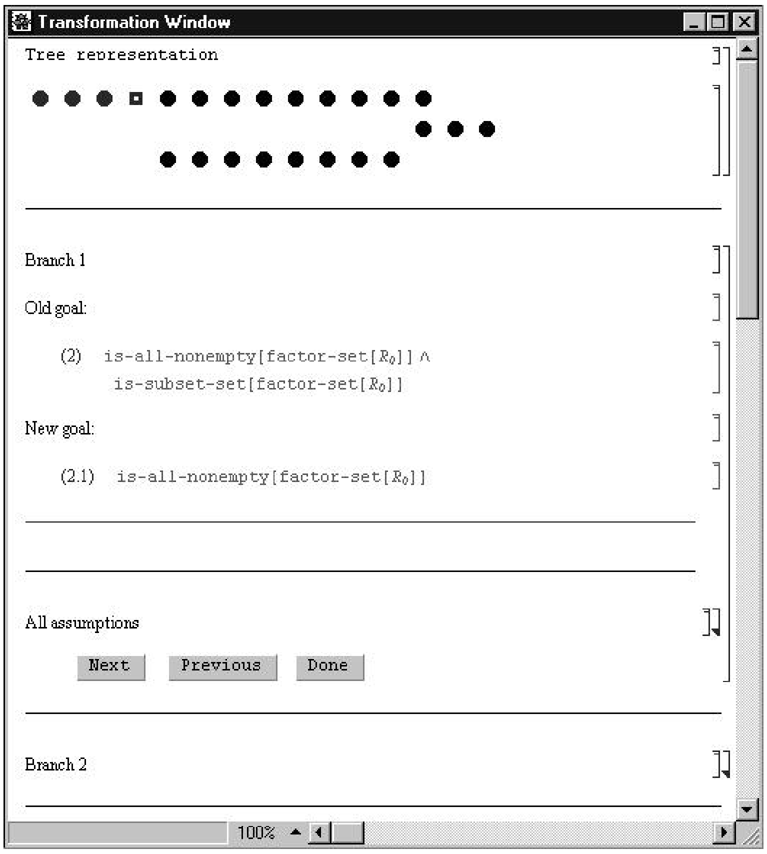
Tree representation

Current goal:

(2) is-all-nonempty[factor-set[R_0]] ∧
is-subset-set[factor-set[R_0]]

All assumptions

Attention Window 4



Tree representation

Branch 1

Old goal:

(2) is-all-nonempty[factor-set[R_0]] ∧
is-subset-set[factor-set[R_0]]

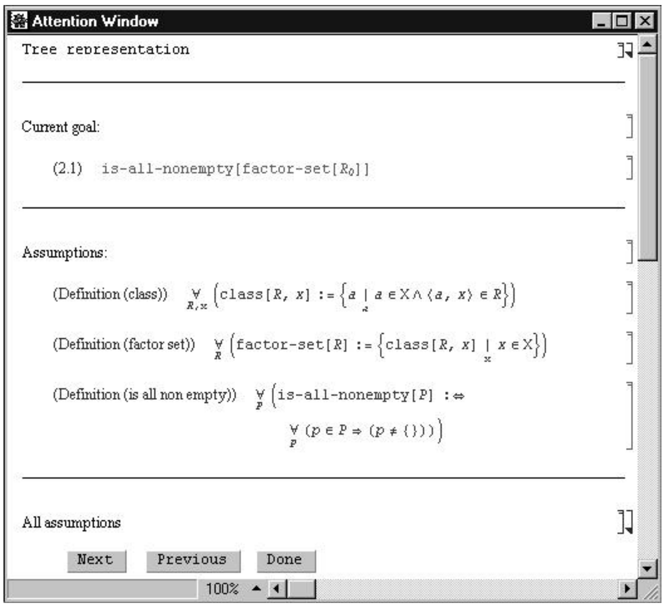
New goal:

(2.1) is-all-nonempty[factor-set[R_0]]

All assumptions

Branch 2

Transformation Window 4



Tree representation

Current goal:

(2.1) is-all-nonempty[factor-set[R_0]]

Assumptions:

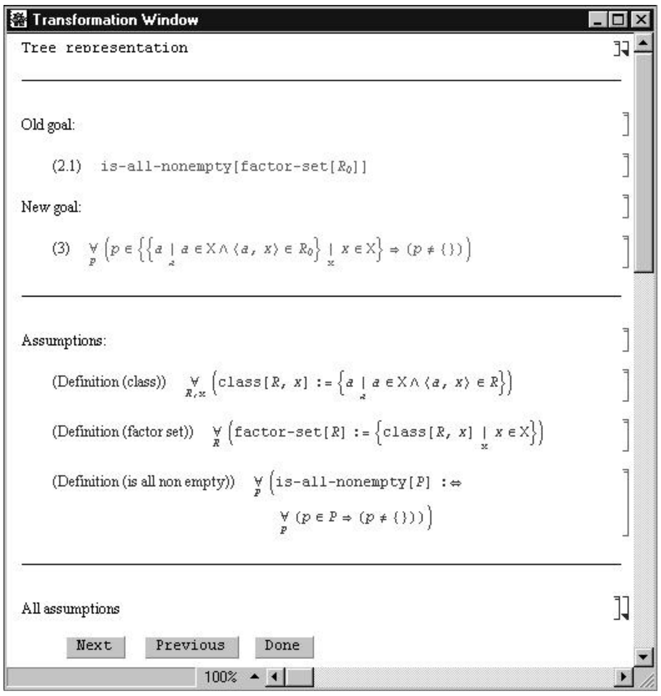
(Definition (class)) $\underset{R,x}{\forall}\left(\text{class}[R,\ x] := \left\{a \underset{a}{|} a \in X \wedge \langle a,\ x\rangle \in R\right\}\right)$

(Definition (factor set)) $\underset{R}{\forall}\left(\text{factor-set}[R] := \left\{\text{class}[R,\ x] \underset{x}{|} x \in X\right\}\right)$

(Definition (is all non empty)) $\underset{P}{\forall}\left(\text{is-all-nonempty}[P] :\Leftrightarrow \underset{p}{\forall}(p \in P \Rightarrow (p \neq \{\}))\right)$

All assumptions

Attention Window 5



Tree representation

Old goal:

(2.1) is-all-nonempty[factor-set[R_0]]

New goal:

(3) $\underset{p}{\forall}\left(p \in \left\{\left\{a \underset{a}{|} a \in X \wedge \langle a,\ x\rangle \in R_0\right\} \underset{x}{|} x \in X\right\} \Rightarrow (p \neq \{\})\right)$

Assumptions:

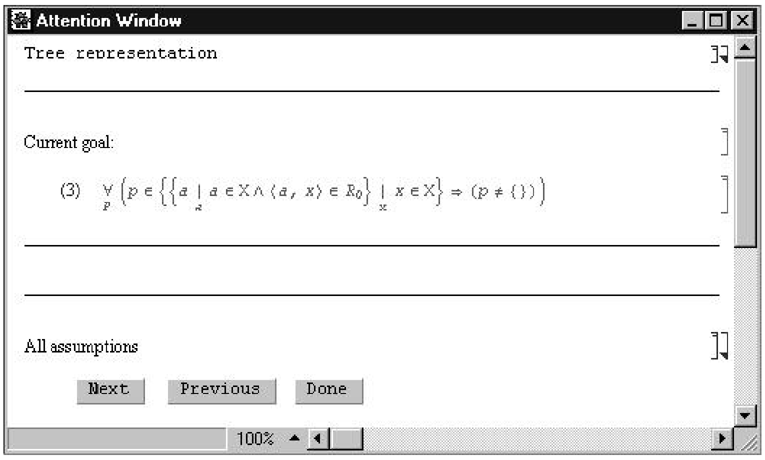
(Definition (class)) $\underset{R,x}{\forall}\left(\text{class}[R,\ x] := \left\{a \underset{a}{|} a \in X \wedge \langle a,\ x\rangle \in R\right\}\right)$

(Definition (factor set)) $\underset{R}{\forall}\left(\text{factor-set}[R] := \left\{\text{class}[R,\ x] \underset{x}{|} x \in X\right\}\right)$

(Definition (is all non empty)) $\underset{P}{\forall}\left(\text{is-all-nonempty}[P] :\Leftrightarrow \underset{p}{\forall}(p \in P \Rightarrow (p \neq \{\}))\right)$

All assumptions

Transformation Window 5



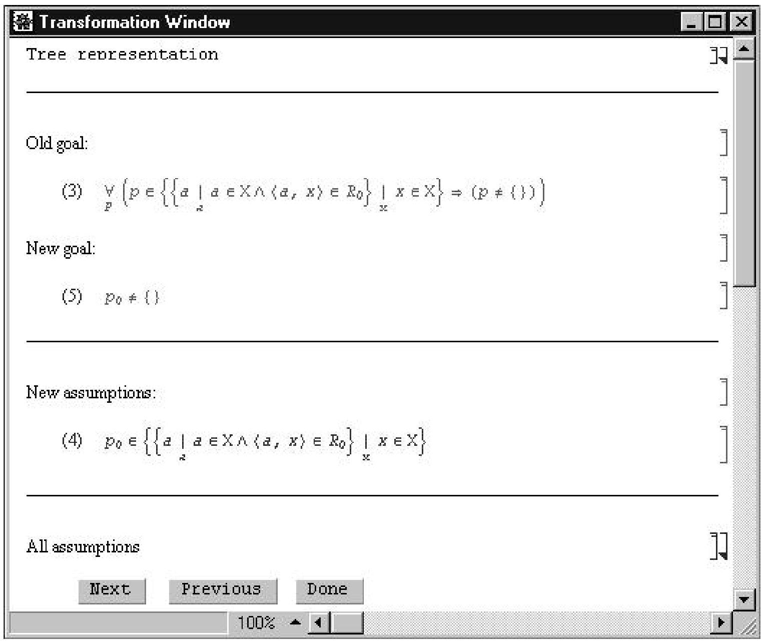
Tree representation

Current goal:

(3) $\forall_{p}\left(p \in \left\{\left\{a \mid_{a} a \in X \wedge \langle a, x\rangle \in R_0\right\} \mid_{x} x \in X\right\} \Rightarrow (p \neq \{\})\right)$

All assumptions

Attention Window 6



Tree representation

Old goal:

(3) $\forall_{p}\left(p \in \left\{\left\{a \mid_{a} a \in X \wedge \langle a, x\rangle \in R_0\right\} \mid_{x} x \in X\right\} \Rightarrow (p \neq \{\})\right)$

New goal:

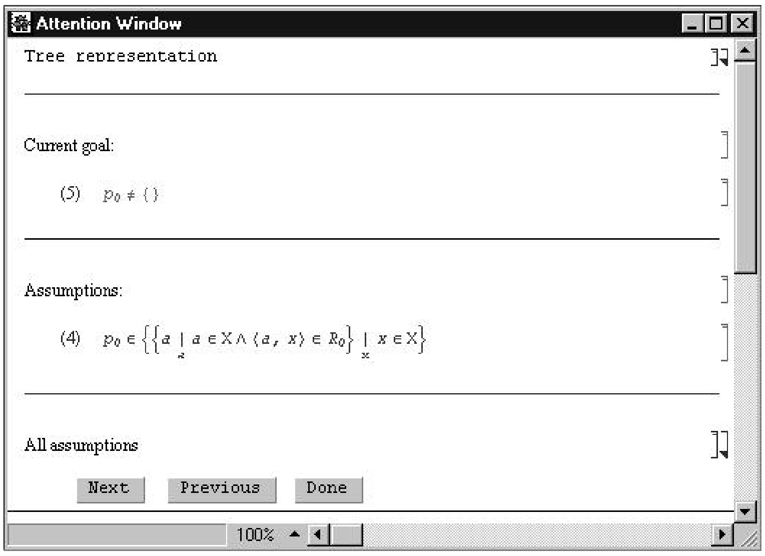
(5) $p_0 \neq \{\}$

New assumptions:

(4) $p_0 \in \left\{\left\{a \mid_{a} a \in X \wedge \langle a, x\rangle \in R_0\right\} \mid_{x} x \in X\right\}$

All assumptions

Transformation Window 6



Tree representation

Current goal:

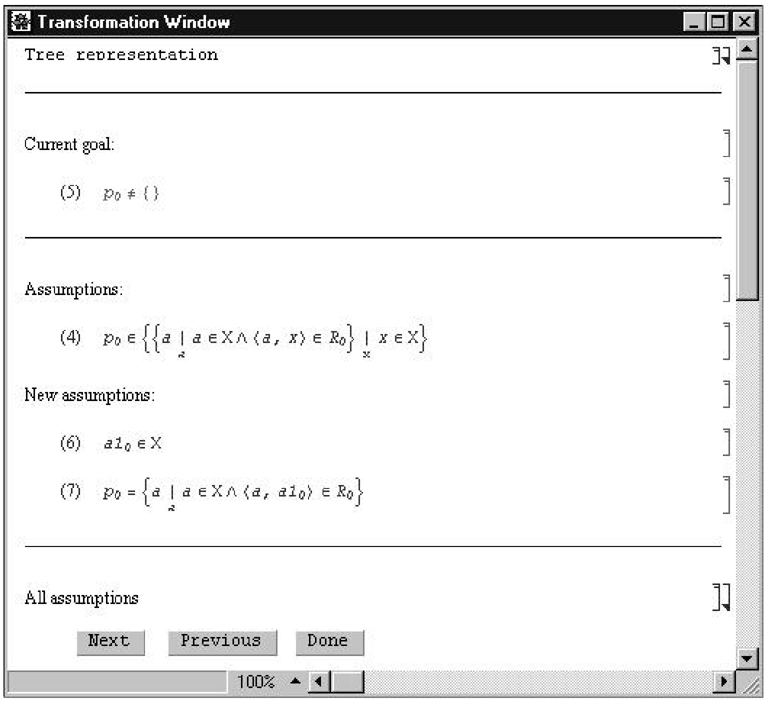
(5) $p_0 \neq \{\}$

Assumptions:

(4) $p_0 \in \{\{a \mid_a a \in X \wedge \langle a, x\rangle \in R_0\} \mid_x x \in X\}$

All assumptions

Attention Window 7



Tree representation

Current goal:

(5) $p_0 \neq \{\}$

Assumptions:

(4) $p_0 \in \{\{a \mid_a a \in X \wedge \langle a, x\rangle \in R_0\} \mid_x x \in X\}$

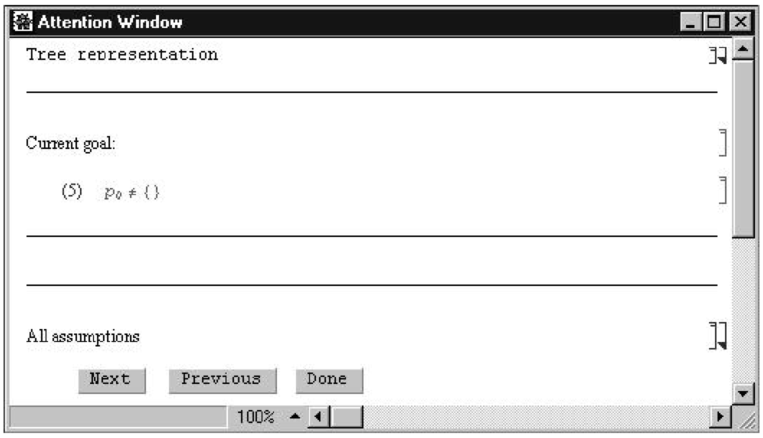
New assumptions:

(6) $a1_0 \in X$

(7) $p_0 = \{a \mid_a a \in X \wedge \langle a, a1_0\rangle \in R_0\}$

All assumptions

Transformation Window 7



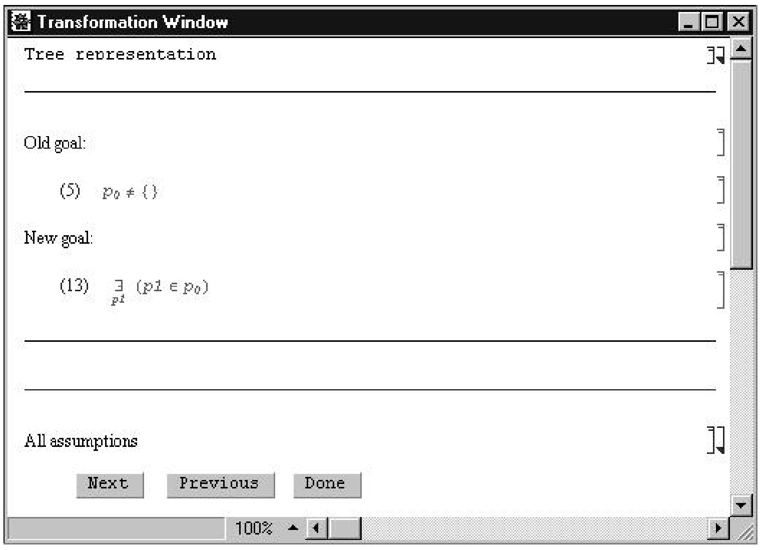
Tree representation

Current goal:

(5) $p_0 \neq \{\}$

All assumptions

Attention Window 8



Tree representation

Old goal:

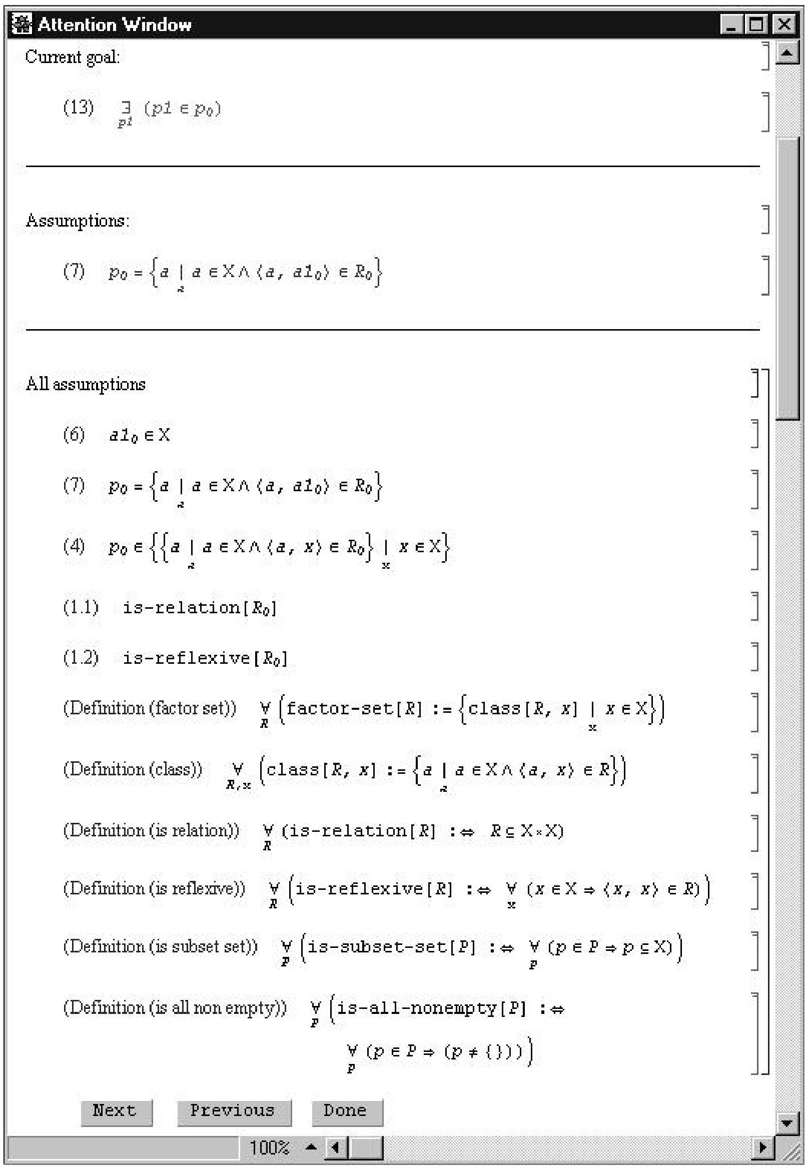
(5) $p_0 \neq \{\}$

New goal:

(13) $\underset{p1}{\exists}\ (p1 \in p_0)$

All assumptions

Transformation Window 8



Current goal:

(13) $\exists_{p1} (p1 \in p_0)$

Assumptions:

(7) $p_0 = \left\{ a \mid_{a} a \in X \wedge \langle a, a1_0 \rangle \in R_0 \right\}$

All assumptions

(6) $a1_0 \in X$

(7) $p_0 = \left\{ a \mid_{a} a \in X \wedge \langle a, a1_0 \rangle \in R_0 \right\}$

(4) $p_0 \in \left\{ \left\{ a \mid_{a} a \in X \wedge \langle a, x \rangle \in R_0 \right\} \mid_{x} x \in X \right\}$

(1.1) $\texttt{is-relation}[R_0]$

(1.2) $\texttt{is-reflexive}[R_0]$

(Definition (factor set)) $\forall_{R} \left(\texttt{factor-set}[R] := \left\{ \texttt{class}[R, x] \mid_{x} x \in X \right\} \right)$

(Definition (class)) $\forall_{R,x} \left(\texttt{class}[R, x] := \left\{ a \mid_{a} a \in X \wedge \langle a, x \rangle \in R \right\} \right)$

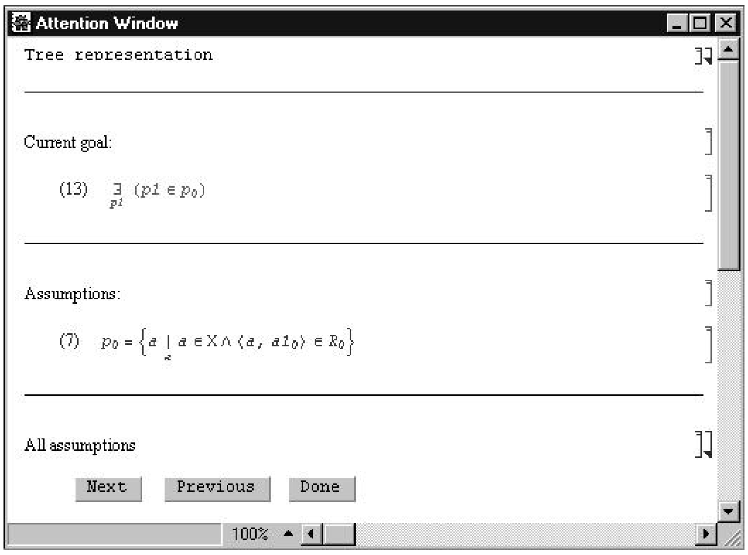
(Definition (is relation)) $\forall_{R} (\texttt{is-relation}[R] :\Leftrightarrow R \subseteq X \times X)$

(Definition (is reflexive)) $\forall_{R} \left(\texttt{is-reflexive}[R] :\Leftrightarrow \forall_{x} (x \in X \Rightarrow \langle x, x \rangle \in R) \right)$

(Definition (is subset set)) $\forall_{P} \left(\texttt{is-subset-set}[P] :\Leftrightarrow \forall_{p} (p \in P \Rightarrow p \subseteq X) \right)$

(Definition (is all non empty)) $\forall_{P} \left(\texttt{is-all-nonempty}[P] :\Leftrightarrow \forall_{p} (p \in P \Rightarrow (p \neq \{\})) \right)$

Attention Window 9 - All assumption area opened



Tree representation

Current goal:

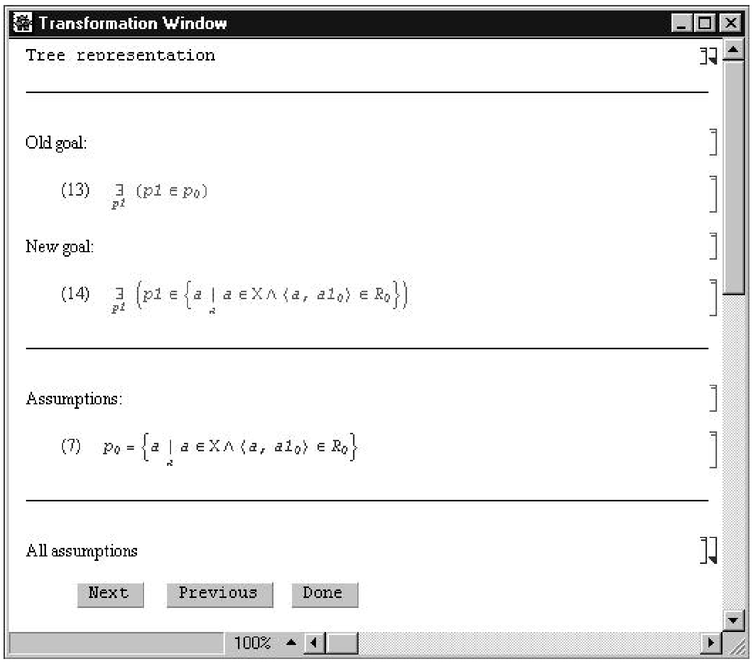
(13) $\exists_{p1} (p1 \in p_0)$

Assumptions:

(7) $p_0 = \left\{a \underset{a}{|} a \in X \wedge \langle a, a1_0 \rangle \in R_0\right\}$

All assumptions

Attention Window 9 - All assumption area closed



Tree representation

Old goal:

(13) $\exists_{p1} (p1 \in p_0)$

New goal:

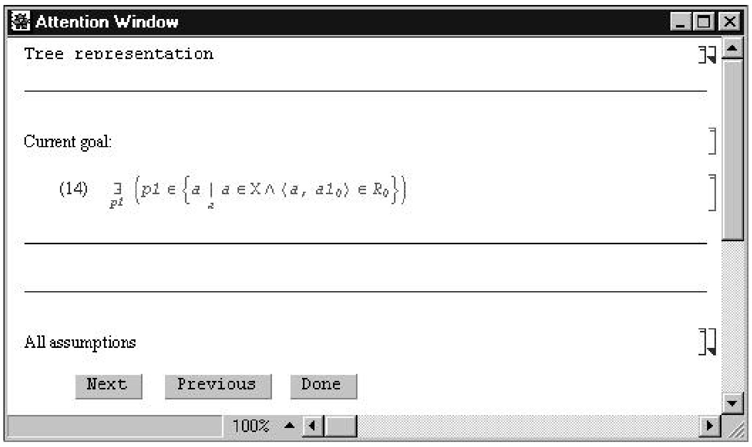
(14) $\exists_{p1} \left(p1 \in \left\{a \underset{a}{|} a \in X \wedge \langle a, a1_0 \rangle \in R_0\right\}\right)$

Assumptions:

(7) $p_0 = \left\{a \underset{a}{|} a \in X \wedge \langle a, a1_0 \rangle \in R_0\right\}$

All assumptions

Transformation Window 9



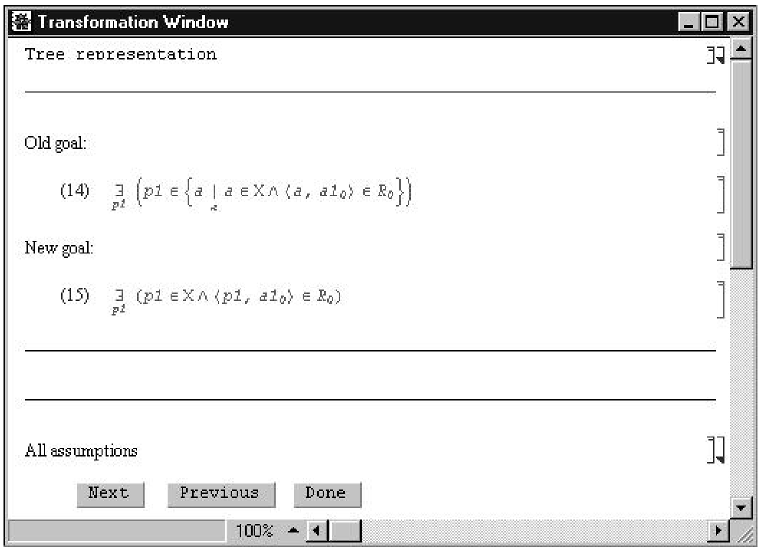
Tree representation

Current goal:

(14) $\exists_{p1}\left(p1 \in \left\{a \mid_{a} a \in X \wedge \langle a, a1_0\rangle \in R_0\right\}\right)$

All assumptions

Attention Window 10



Tree representation

Old goal:

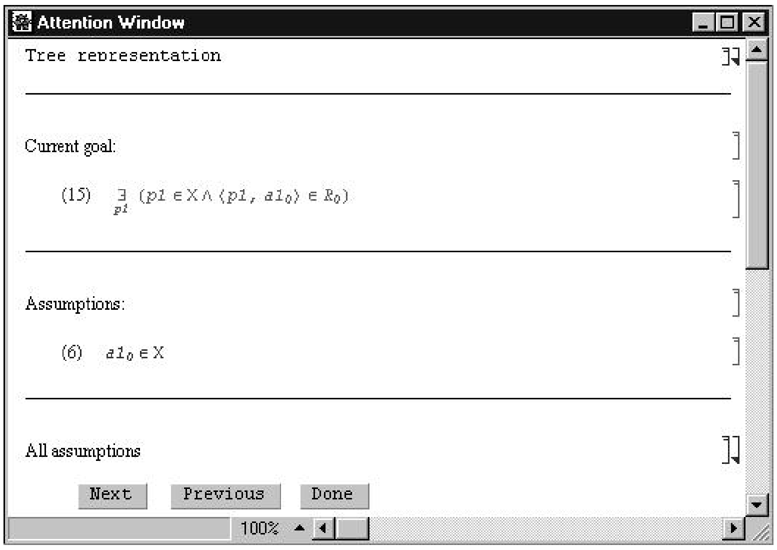
(14) $\exists_{p1}\left(p1 \in \left\{a \mid_{a} a \in X \wedge \langle a, a1_0\rangle \in R_0\right\}\right)$

New goal:

(15) $\exists_{p1}\left(p1 \in X \wedge \langle p1, a1_0\rangle \in R_0\right)$

All assumptions

Transformation Window 10



Tree representation

Current goal:

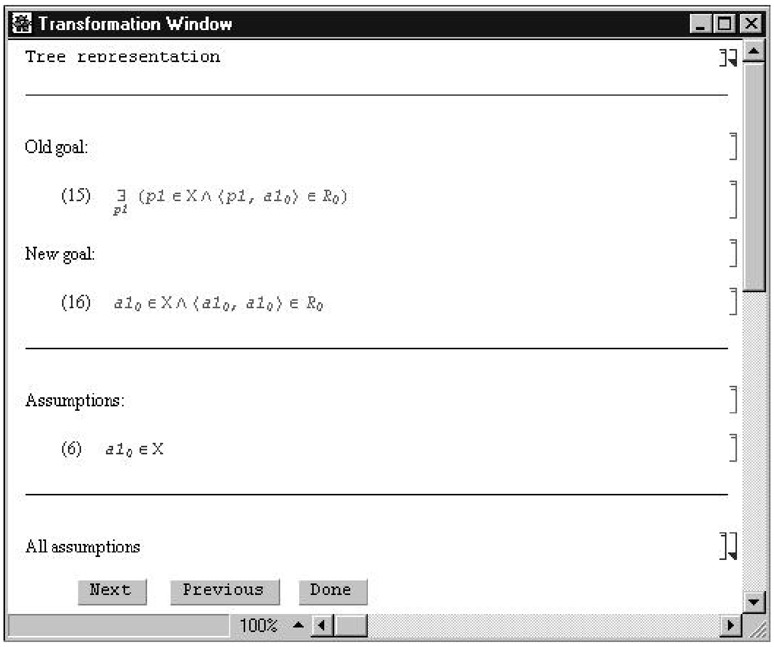
(15) $\underset{p1}{\exists}\ (p1 \in X \land \langle p1, a1_0\rangle \in R_0)$

Assumptions:

(6) $a1_0 \in X$

All assumptions

Attention Window 11



Tree representation

Old goal:

(15) $\underset{p1}{\exists}\ (p1 \in X \land \langle p1, a1_0\rangle \in R_0)$

New goal:

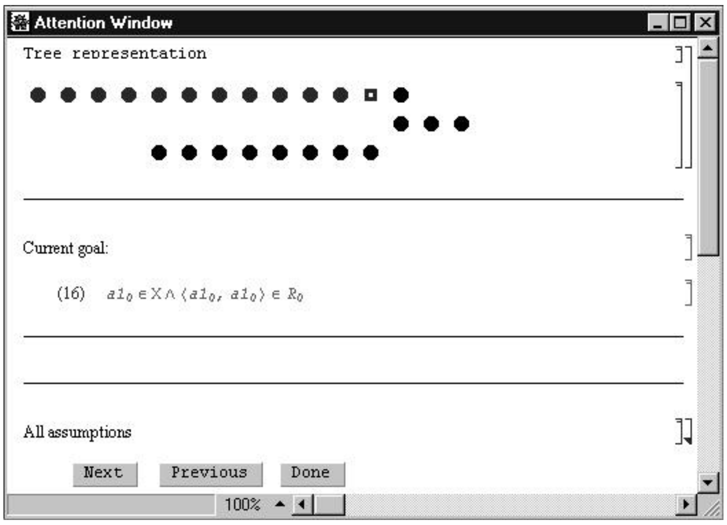
(16) $a1_0 \in X \land \langle a1_0, a1_0\rangle \in R_0$

Assumptions:

(6) $a1_0 \in X$

All assumptions

Transformation Window 11

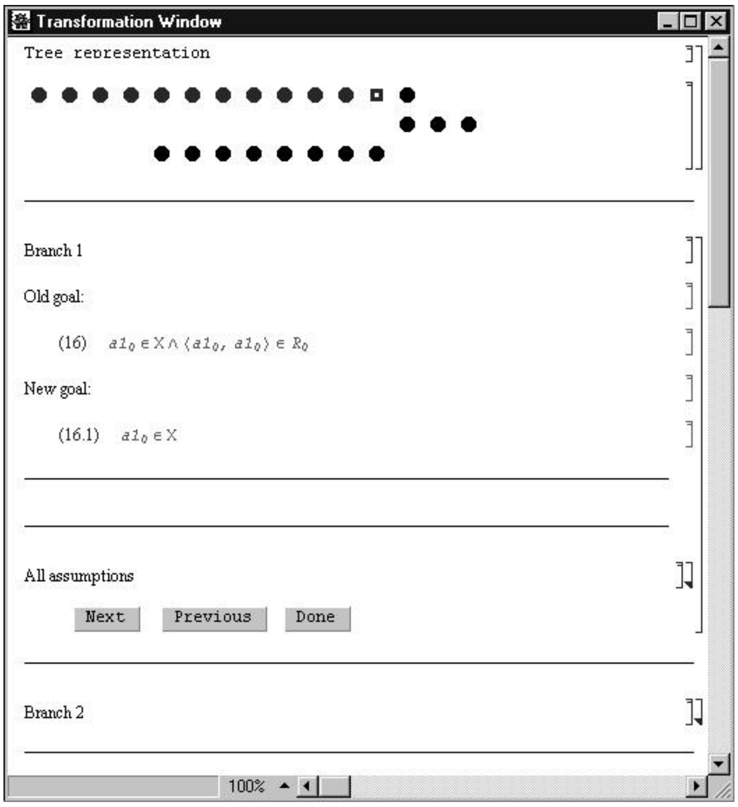

Tree representation

Current goal:

(16) $a1_0 \in X \wedge \langle a1_0, a1_0 \rangle \in R_0$

All assumptions

Attention Window 12


Tree representation

Branch 1

Old goal:

(16) $a1_0 \in X \wedge \langle a1_0, a1_0 \rangle \in R_0$

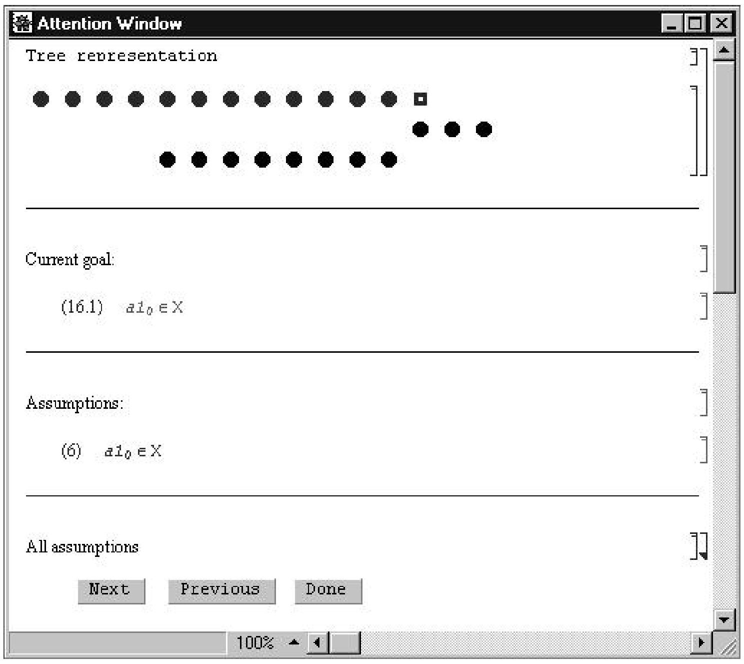
New goal:

(16.1) $a1_0 \in X$

All assumptions

Branch 2

Transformation Window 12



Current goal:

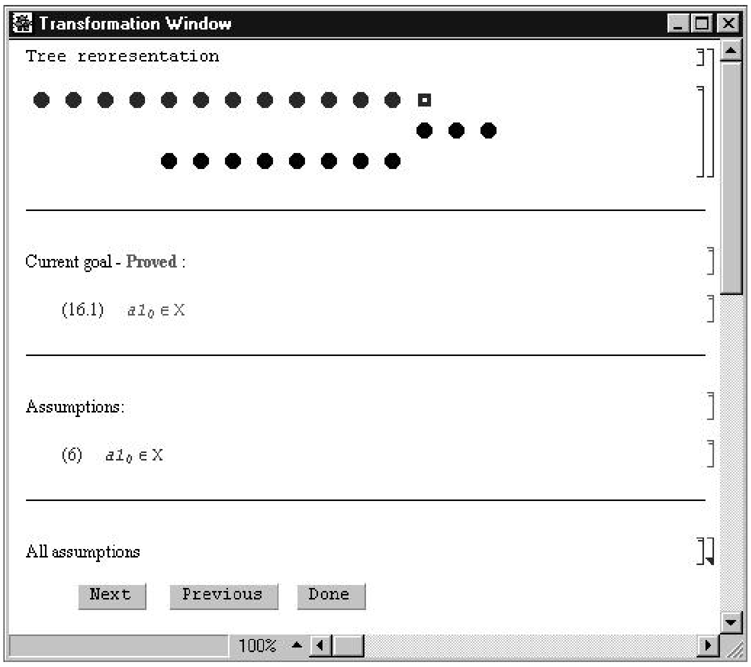
(16.1) $a1_0 \in X$

Assumptions:

(6) $a1_0 \in X$

All assumptions

Attention Window 13



Current goal - **Proved** :

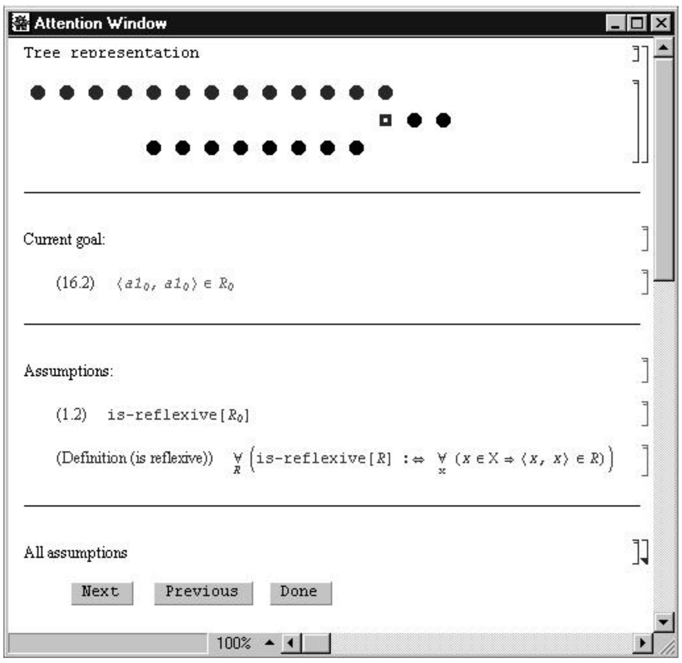
(16.1) $a1_0 \in X$

Assumptions:

(6) $a1_0 \in X$

All assumptions

Transformation Window 13



Tree representation

Current goal:

(16.2) $\langle a1_0, a1_0 \rangle \in R_0$

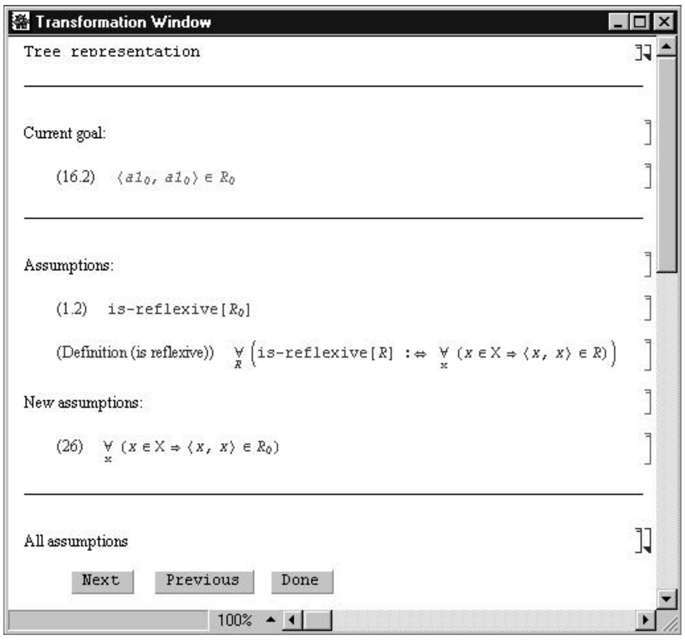
Assumptions:

(1.2) is-reflexive[R_0]

(Definition (is reflexive)) $\forall_{R} \left(\text{is-reflexive}[R] :\Leftrightarrow \forall_{x} (x \in X \Rightarrow \langle x, x \rangle \in R) \right)$

All assumptions

Attention Window 14



Tree representation

Current goal:

(16.2) $\langle a1_0, a1_0 \rangle \in R_0$

Assumptions:

(1.2) is-reflexive[R_0]

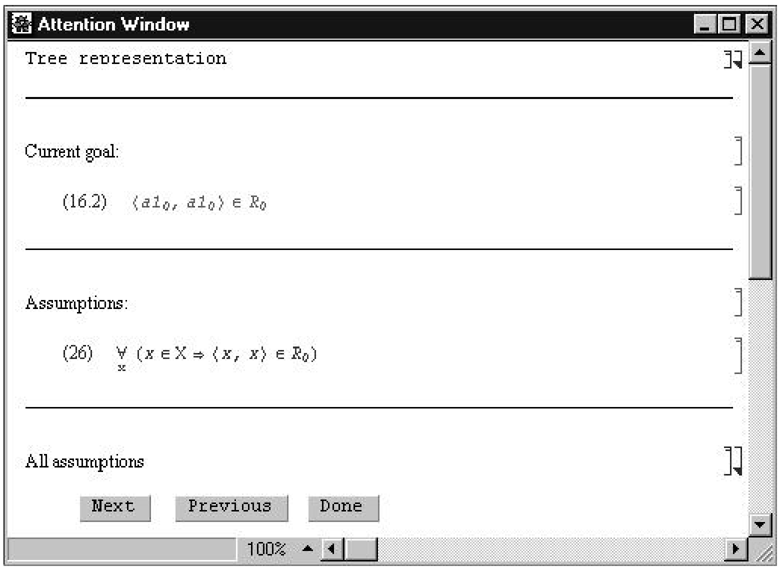
(Definition (is reflexive)) $\forall_{R} \left(\text{is-reflexive}[R] :\Leftrightarrow \forall_{x} (x \in X \Rightarrow \langle x, x \rangle \in R) \right)$

New assumptions:

(26) $\forall_{x} (x \in X \Rightarrow \langle x, x \rangle \in R_0)$

All assumptions

Transformation Window 14



Tree representation

Current goal:

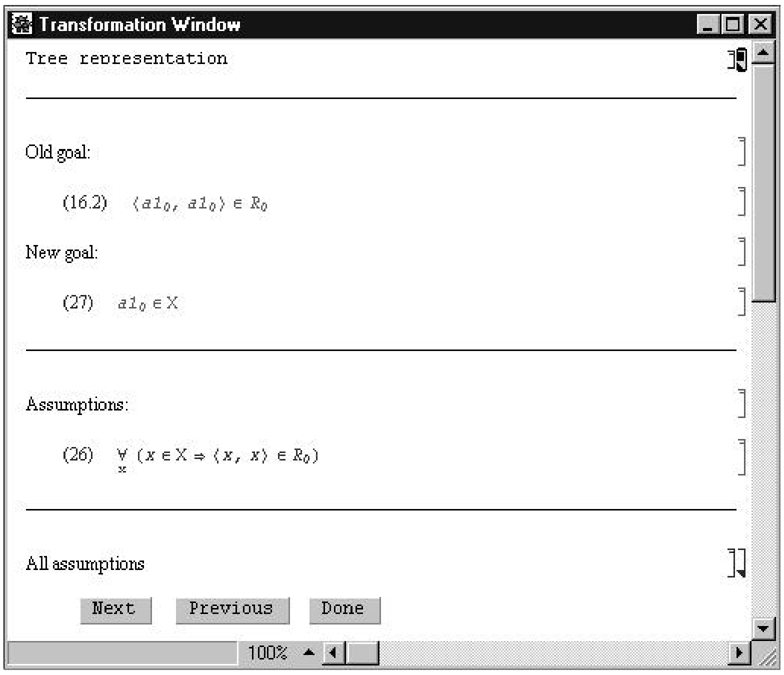
(16.2) $\langle a1_0, a1_0\rangle \in R_0$

Assumptions:

(26) $\forall_x (x \in X \Rightarrow \langle x, x\rangle \in R_0)$

All assumptions

Attention Window 15



Tree representation

Old goal:

(16.2) $\langle a1_0, a1_0\rangle \in R_0$

New goal:

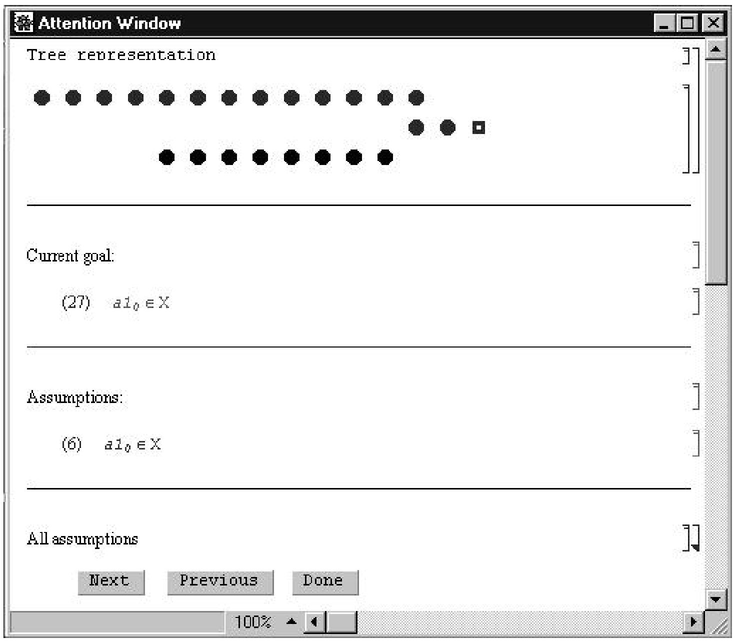
(27) $a1_0 \in X$

Assumptions:

(26) $\forall_x (x \in X \Rightarrow \langle x, x\rangle \in R_0)$

All assumptions

Transformation Window 15



Current goal:

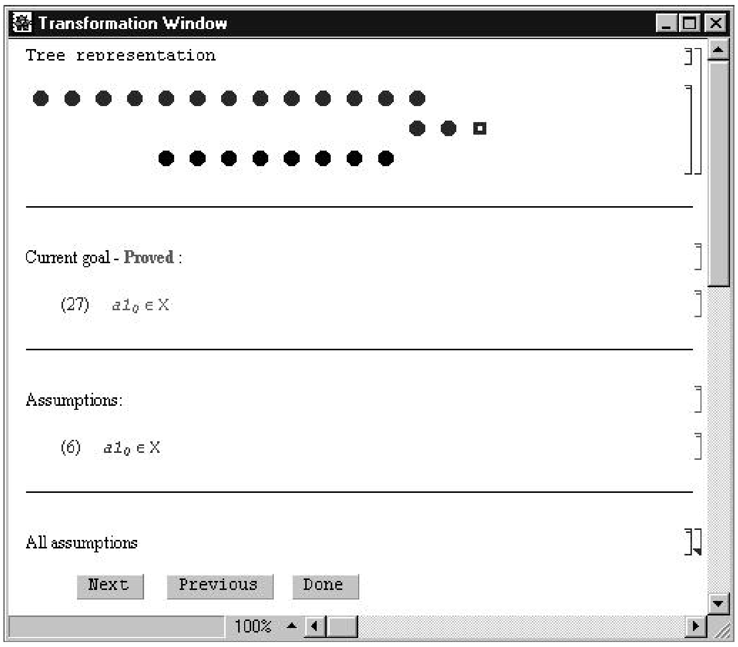
(27) $a1_0 \in X$

Assumptions:

(6) $a1_0 \in X$

All assumptions

Attention Window 16



Current goal - **Proved** :

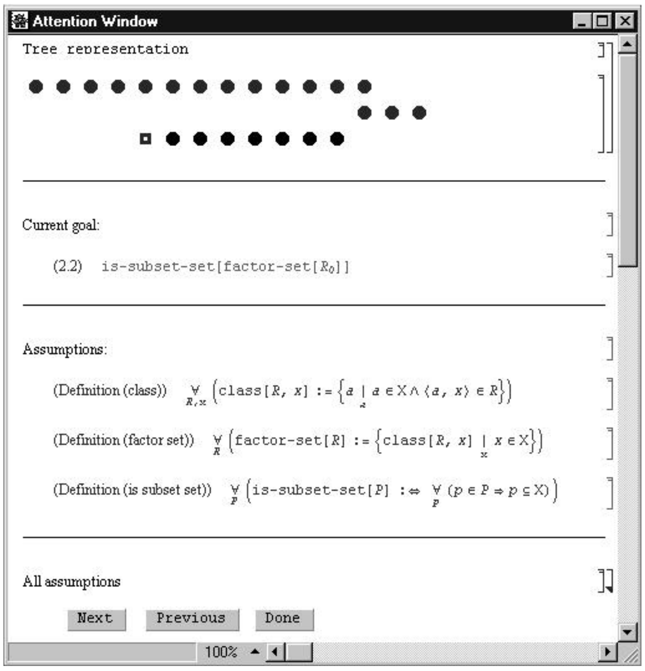
(27) $a1_0 \in X$

Assumptions:

(6) $a1_0 \in X$

All assumptions

Transformation Window 16



Tree representation

Current goal:

(2.2) is-subset-set[factor-set[R_0]]

Assumptions:

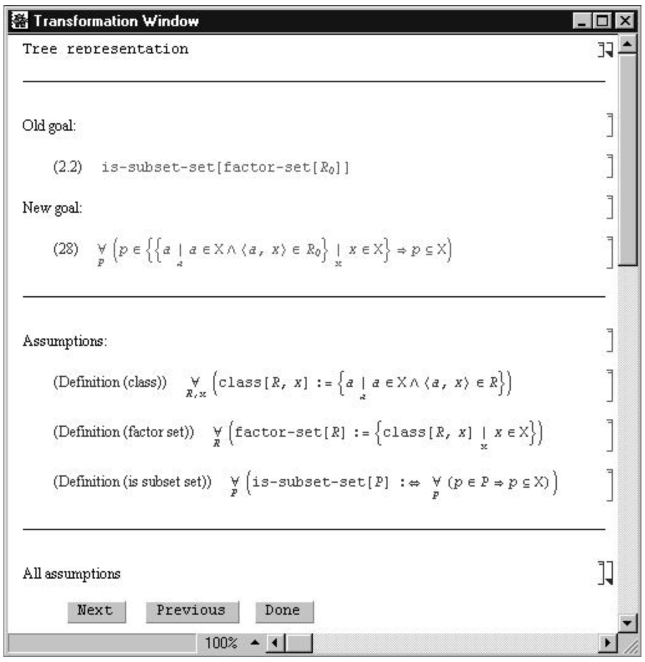
(Definition (class)) $\underset{R,x}{\forall}\left(\text{class}[R, x] := \left\{a \underset{a}{|} a \in X \wedge \langle a, x\rangle \in R\right\}\right)$

(Definition (factor set)) $\underset{R}{\forall}\left(\text{factor-set}[R] := \left\{\text{class}[R, x] \underset{x}{|} x \in X\right\}\right)$

(Definition (is subset set)) $\underset{P}{\forall}\left(\text{is-subset-set}[P] :\Leftrightarrow \underset{p}{\forall}(p \in P \Rightarrow p \subseteq X)\right)$

All assumptions

Attention Window 17



Tree representation

Old goal:

(2.2) is-subset-set[factor-set[R_0]]

New goal:

(28) $\underset{p}{\forall}\left(p \in \left\{\left\{a \underset{a}{|} a \in X \wedge \langle a, x\rangle \in R_0\right\} \underset{x}{|} x \in X\right\} \Rightarrow p \subseteq X\right)$

Assumptions:

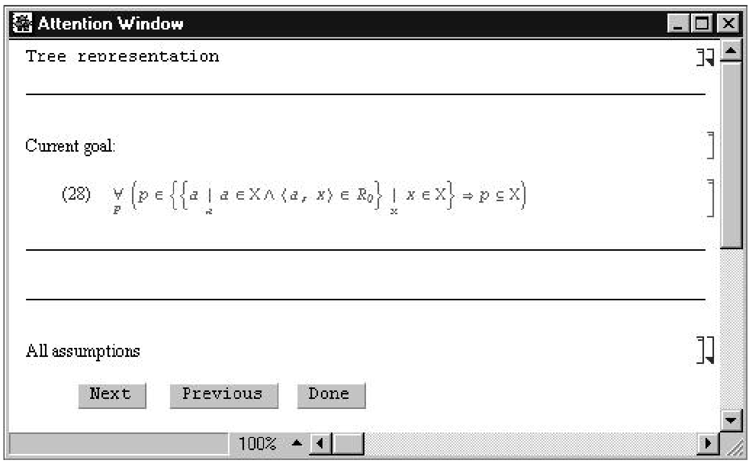
(Definition (class)) $\underset{R,x}{\forall}\left(\text{class}[R, x] := \left\{a \underset{a}{|} a \in X \wedge \langle a, x\rangle \in R\right\}\right)$

(Definition (factor set)) $\underset{R}{\forall}\left(\text{factor-set}[R] := \left\{\text{class}[R, x] \underset{x}{|} x \in X\right\}\right)$

(Definition (is subset set)) $\underset{P}{\forall}\left(\text{is-subset-set}[P] :\Leftrightarrow \underset{p}{\forall}(p \in P \Rightarrow p \subseteq X)\right)$

All assumptions

Transformation Window 17



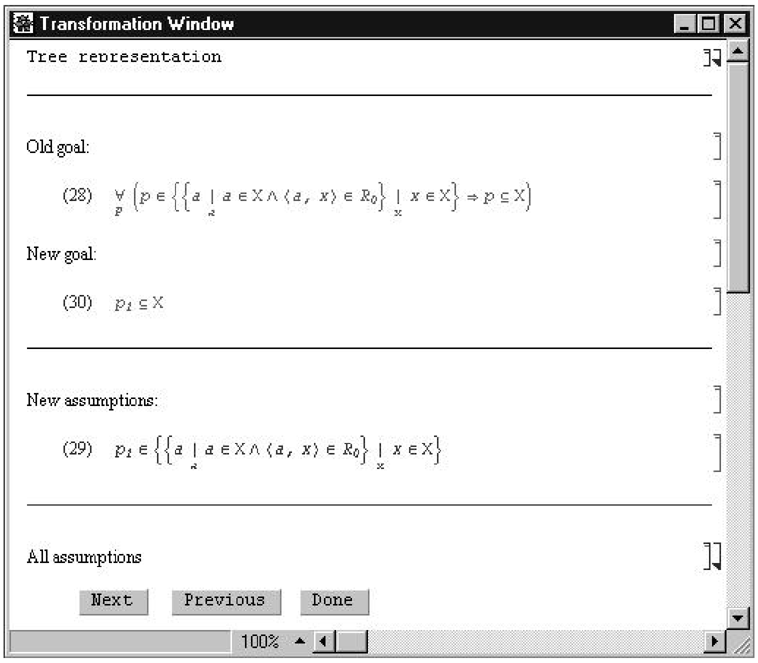
Tree representation

Current goal:

(28) $\underset{p}{\forall}\left(p \in \left\{\left\{a \underset{a}{\mid} a \in X \wedge \langle a, x\rangle \in R_0\right\} \underset{x}{\mid} x \in X\right\} \Rightarrow p \subseteq X\right)$

All assumptions

Attention Window 18



Tree representation

Old goal:

(28) $\underset{p}{\forall}\left(p \in \left\{\left\{a \underset{a}{\mid} a \in X \wedge \langle a, x\rangle \in R_0\right\} \underset{x}{\mid} x \in X\right\} \Rightarrow p \subseteq X\right)$

New goal:

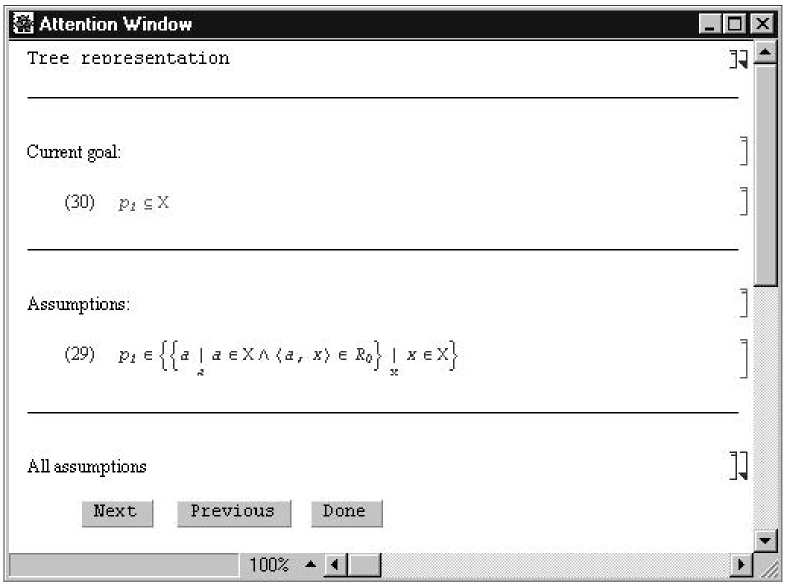
(30) $p_1 \subseteq X$

New assumptions:

(29) $p_1 \in \left\{\left\{a \underset{a}{\mid} a \in X \wedge \langle a, x\rangle \in R_0\right\} \underset{x}{\mid} x \in X\right\}$

All assumptions

Transformation Window 18



Tree representation

Current goal:

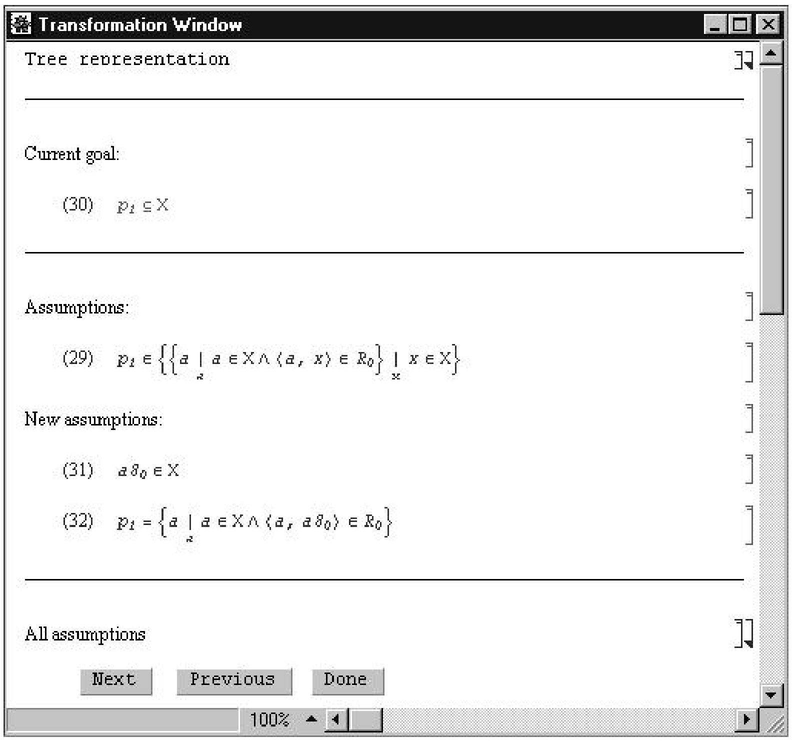
(30) $p_1 \subseteq X$

Assumptions:

(29) $p_1 \in \left\{\left\{a \underset{a}{|} a \in X \wedge \langle a, x\rangle \in R_0\right\} \underset{x}{|} x \in X\right\}$

All assumptions

Attention Window 19



Tree representation

Current goal:

(30) $p_1 \subseteq X$

Assumptions:

(29) $p_1 \in \left\{\left\{a \underset{a}{|} a \in X \wedge \langle a, x\rangle \in R_0\right\} \underset{x}{|} x \in X\right\}$

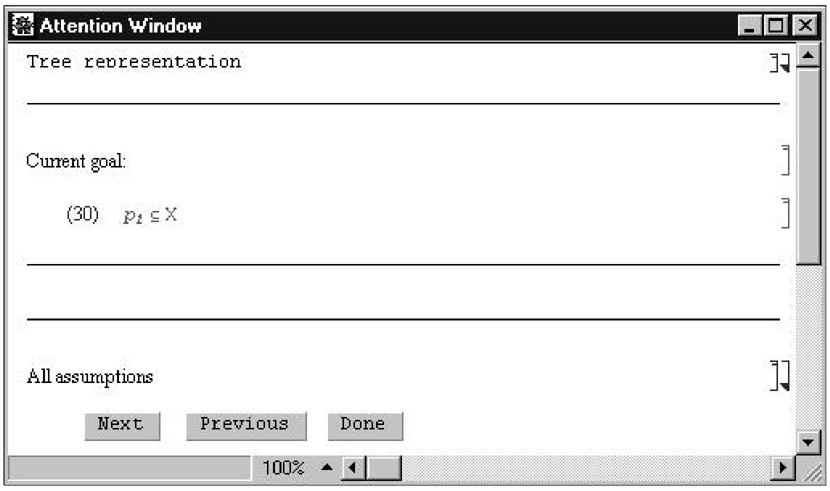
New assumptions:

(31) $a\delta_0 \in X$

(32) $p_1 = \left\{a \underset{a}{|} a \in X \wedge \langle a, a\delta_0\rangle \in R_0\right\}$

All assumptions

Transformation Window 19



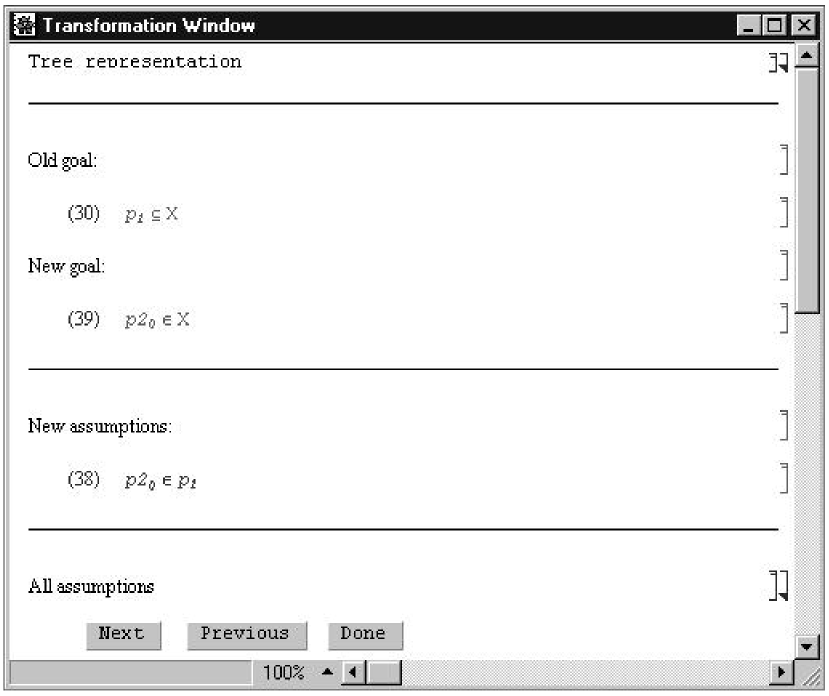
Tree representation

Current goal:

(30) $p_1 \subseteq X$

All assumptions

Attention Window 20



Tree representation

Old goal:

(30) $p_1 \subseteq X$

New goal:

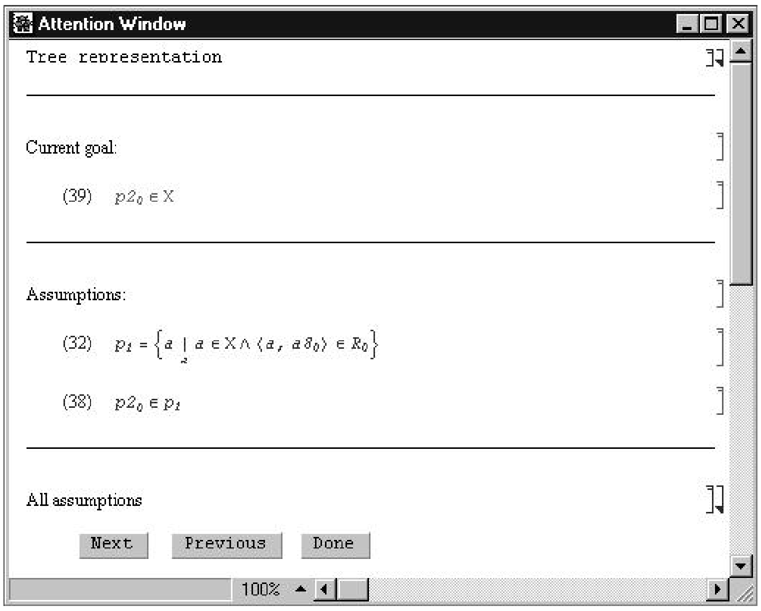
(39) $p2_0 \in X$

New assumptions:

(38) $p2_0 \in p_1$

All assumptions

Transformation Window 20



Tree representation

Current goal:

(39) $p2_0 \in X$

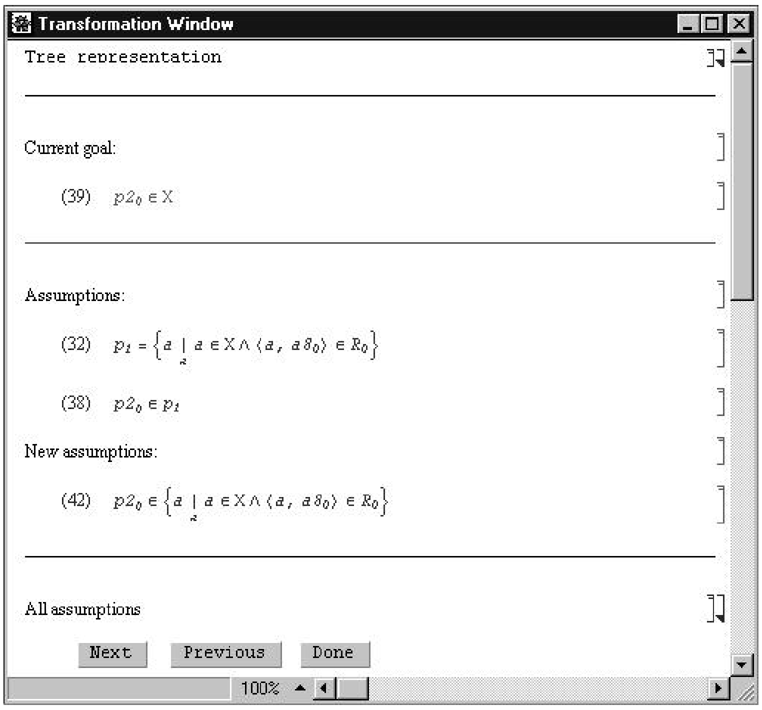
Assumptions:

(32) $p_1 = \left\{ a \underset{a}{|} a \in X \wedge \langle a, a\delta_0 \rangle \in R_0 \right\}$

(38) $p2_0 \in p_1$

All assumptions

Attention Window 21



Tree representation

Current goal:

(39) $p2_0 \in X$

Assumptions:

(32) $p_1 = \left\{ a \underset{a}{|} a \in X \wedge \langle a, a\delta_0 \rangle \in R_0 \right\}$

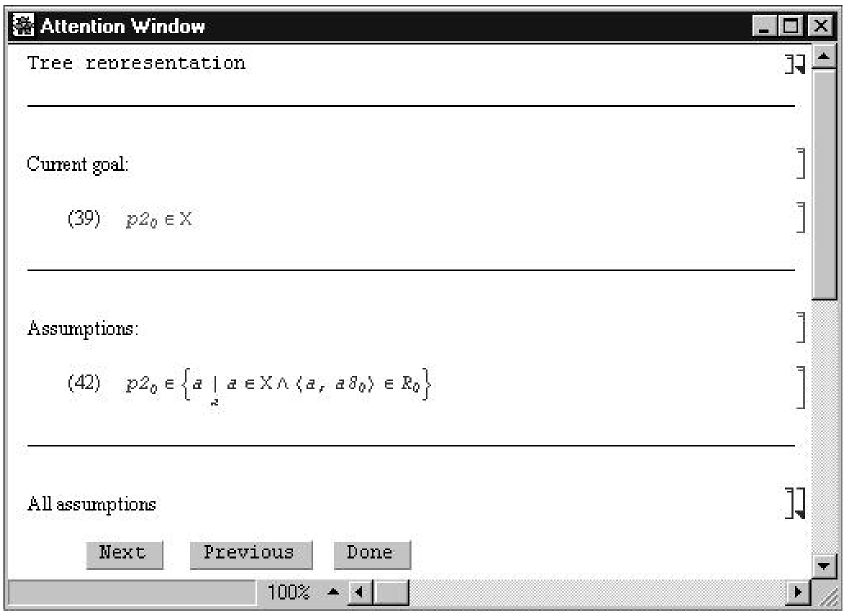
(38) $p2_0 \in p_1$

New assumptions:

(42) $p2_0 \in \left\{ a \underset{a}{|} a \in X \wedge \langle a, a\delta_0 \rangle \in R_0 \right\}$

All assumptions

Transformation Window 21



Tree representation

Current goal:

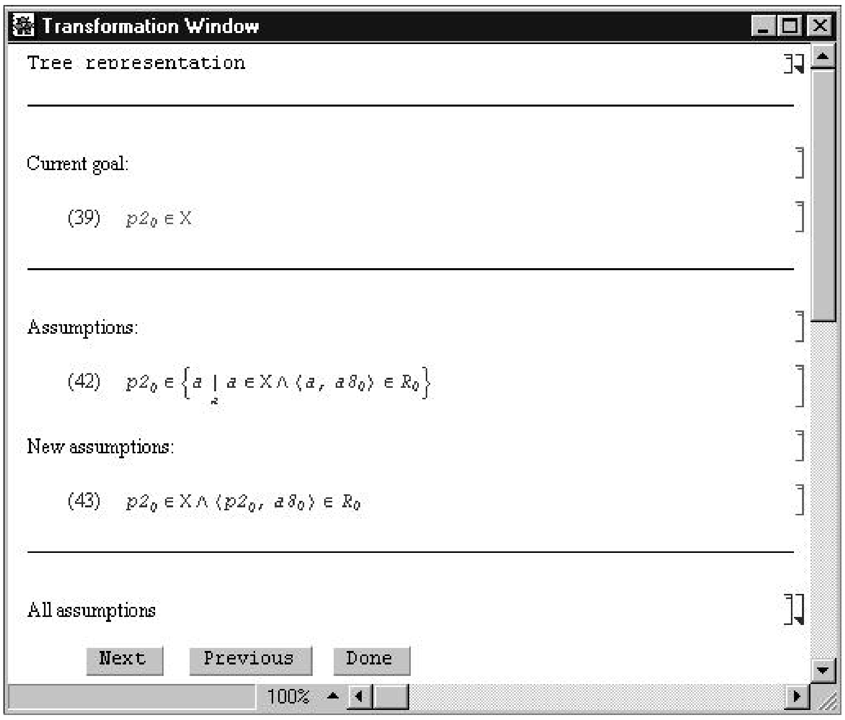
(39) $p2_0 \in X$

Assumptions:

(42) $p2_0 \in \{a \mid_a a \in X \wedge \langle a, a8_0 \rangle \in R_0\}$

All assumptions

Attention Window 22



Tree representation

Current goal:

(39) $p2_0 \in X$

Assumptions:

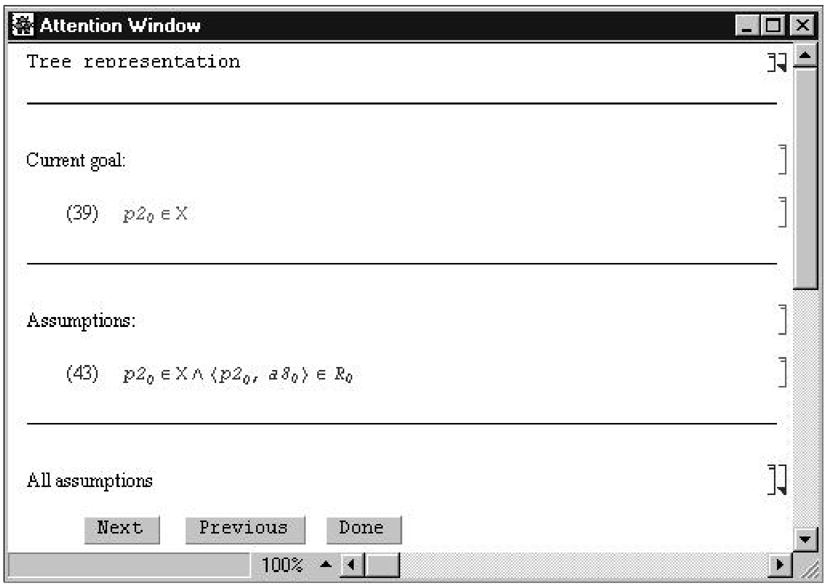
(42) $p2_0 \in \{a \mid_a a \in X \wedge \langle a, a8_0 \rangle \in R_0\}$

New assumptions:

(43) $p2_0 \in X \wedge \langle p2_0, a8_0 \rangle \in R_0$

All assumptions

Transformation Window 22


Attention Window

Tree representation

Current goal:

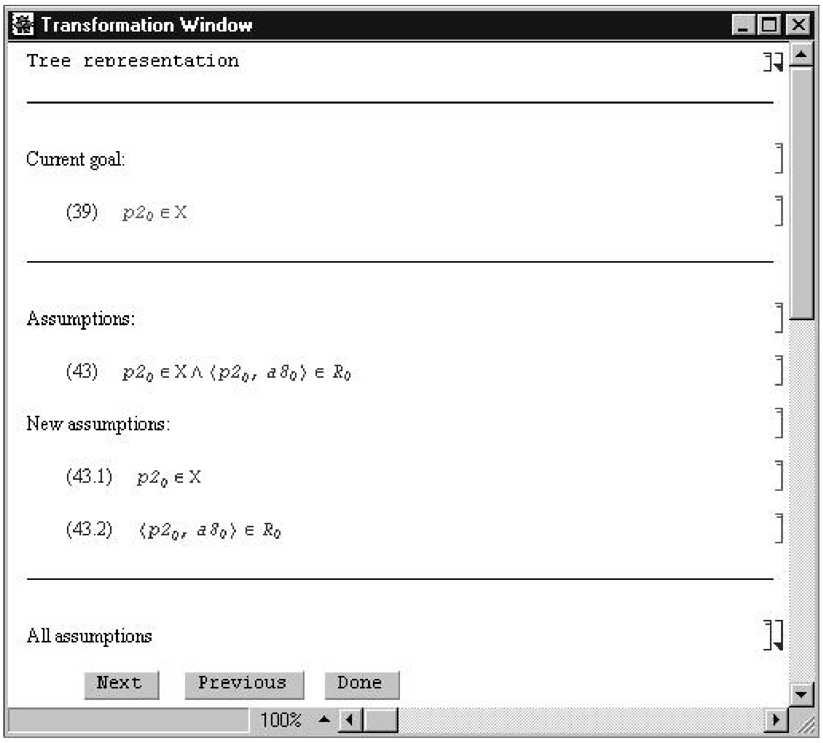
(39) $p2_0 \in X$

Assumptions:

(43) $p2_0 \in X \wedge \langle p2_0, a8_0 \rangle \in R_0$

All assumptions

Attention Window 23


Transformation Window

Tree representation

Current goal:

(39) $p2_0 \in X$

Assumptions:

(43) $p2_0 \in X \wedge \langle p2_0, a8_0 \rangle \in R_0$

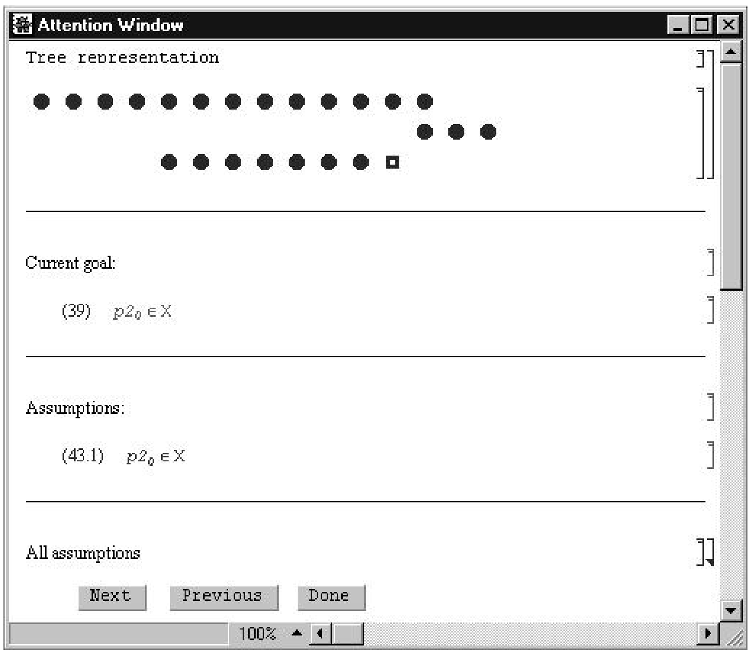
New assumptions:

(43.1) $p2_0 \in X$

(43.2) $\langle p2_0, a8_0 \rangle \in R_0$

All assumptions

Transformation Window 23



Tree representation

Current goal:

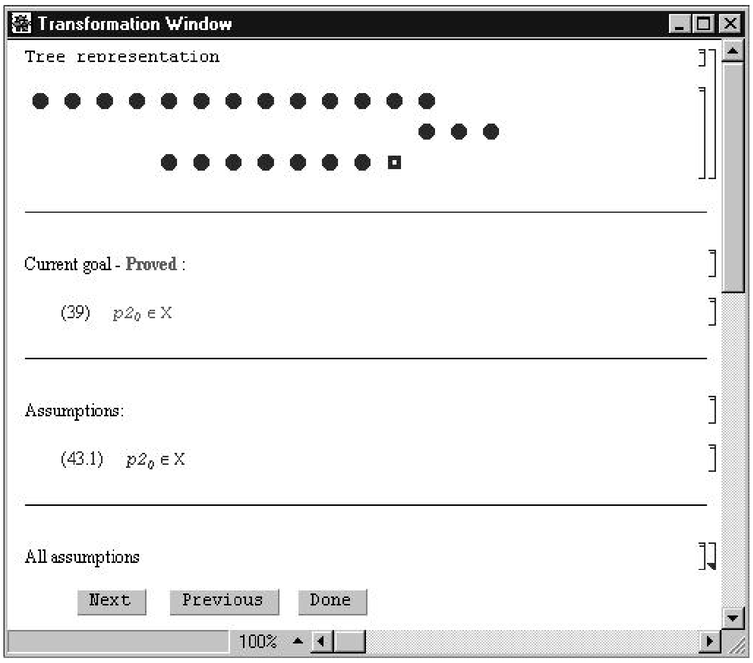
(39) $p2_0 \in X$

Assumptions:

(43.1) $p2_0 \in X$

All assumptions

Attention Window 24



Tree representation

Current goal - **Proved** :

(39) $p2_0 \in X$

Assumptions:

(43.1) $p2_0 \in X$

All assumptions

Transformation Window 24