

# Theory Plug-in v0.5 User Manual

This document provides the user manual of the Theory Plug-in v0.5. The Theory plug-in supersedes the Rule-based Prover v0.3. The Theory plug-in comes with an upgraded version of the Rule-based Prover (we shall refer to from here on as RbP) that works with mathematical extensions.

## Overview

The Theory plug-in is a Rodin extension that provides the facility to define mathematical extensions as well as prover extensions.

Mathematical extensions are new operator definitions and new datatype definitions. Operator definitions can be expression operators (card) and predicate operators (finite). Datatypes extensions can be used to define enumerated datatypes (DIRECTION) as well as inductive datatypes (Tree).

The placeholder for mathematical and prover extensions is a Theory construct which looks similar to contexts and machines. A theory can include datatypes definitions, operator definitions, inference and rewrite rules. In this user manual, we provide a step-by-step guide to developing and using theories.

For installation details of the Theory plug-in, please refer to the Wiki page of the plug-in that can be found at: [http://wiki.event-b.org/index.php/Theory\\_Plug-in](http://wiki.event-b.org/index.php/Theory_Plug-in)

## Theory Development

### Create a New Theory

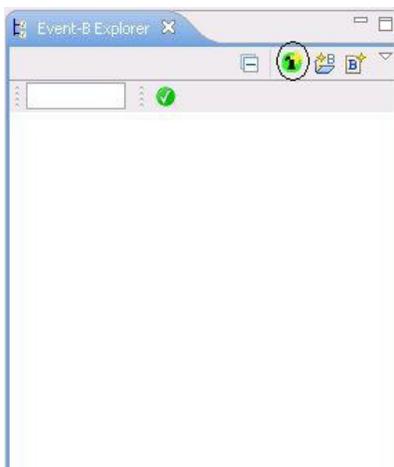


Figure 1. Create a new theory

When the Theory plug-in is installed, an additional action is added to the Event-B Explorer. In Figure 1, the circled button provides the ability to create a new theory in a designated project called “**MathExtensions**”. Theories have file extension “.tuf”, as in *Theory Unchecked File*.

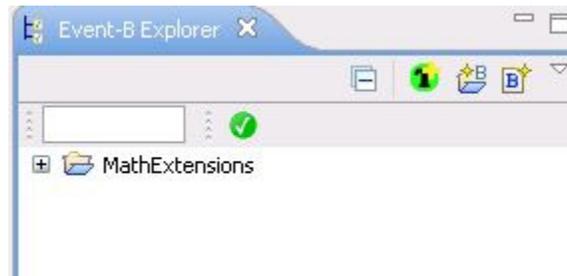


Figure 2. MathExtensions project

To create a theory, follow the following steps:

- 1- Click on the  in the Event-B Explorer.
- 2- You will see the following dialog:

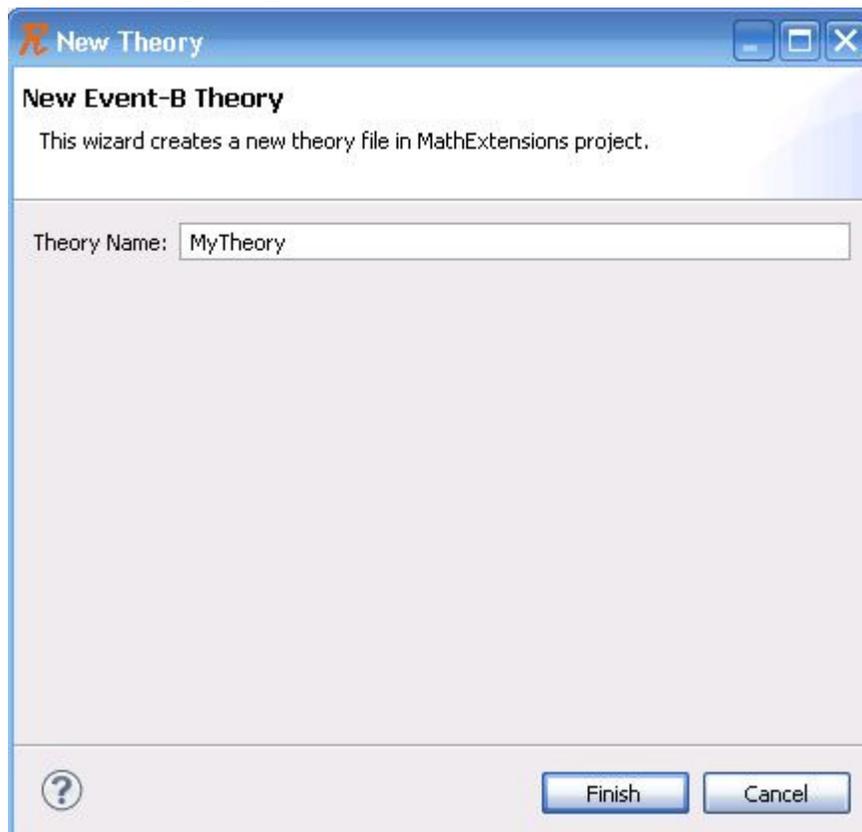


Figure 3. New Theory Dialog

3- Select the theory name in the provided text field, and click finish.

Alternatively, theories can be created as follows:

- 1- Right-click on any Event-B project in the Event-B Explorer.
- 2- Follow New -> Other -> Event-B and choose "Event-B Theory".
- 3- The dialog in Figure 3. Will be shown. Follow the same instructions as before to create a theory.

The Theory editor will show up as follows:

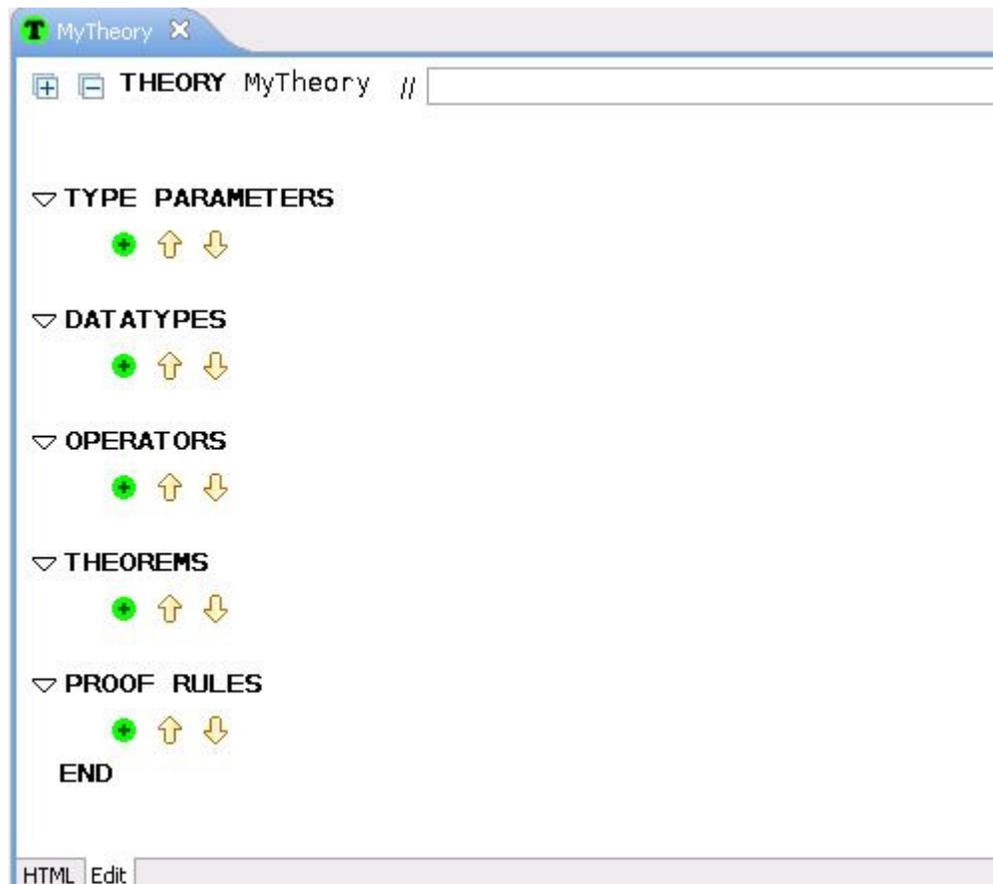


Figure 4. Theory Editor

The use of the theory editor is similar to that of context and machine editors. The theory editor is fitted with an HTML viewer (pretty printer).

The Event-B Explorer is extended to display unchecked theories:

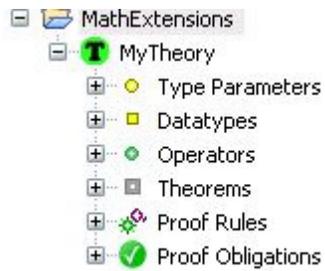


Figure 5. Explorer with Theories

### Add Type Parameters

Type parameters are used to parameterize the theory, and consequently provide polymorphism.

To add a type parameter, click on  under Type Parameters heading.

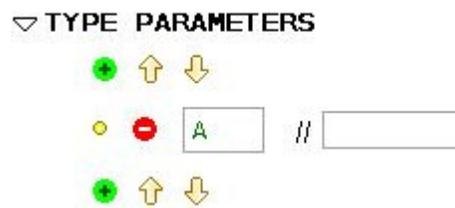


Figure 6. Add Type Parameter

### Add Datatype Definitions

Datatypes can be added by clicking on  under Datatypes heading.

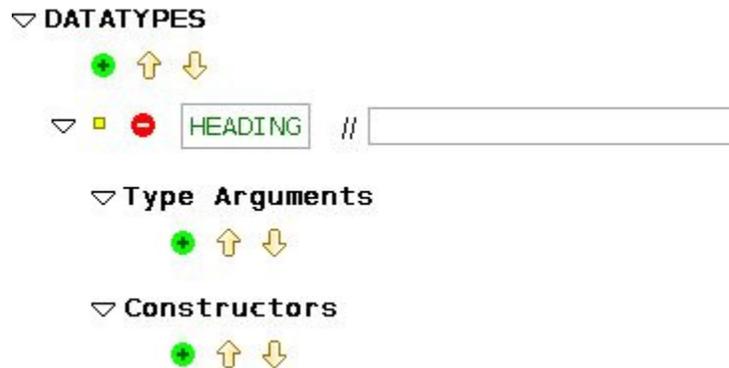


Figure 7. Add Datatype

A complete definition of the HEADING datatype can be as follows:

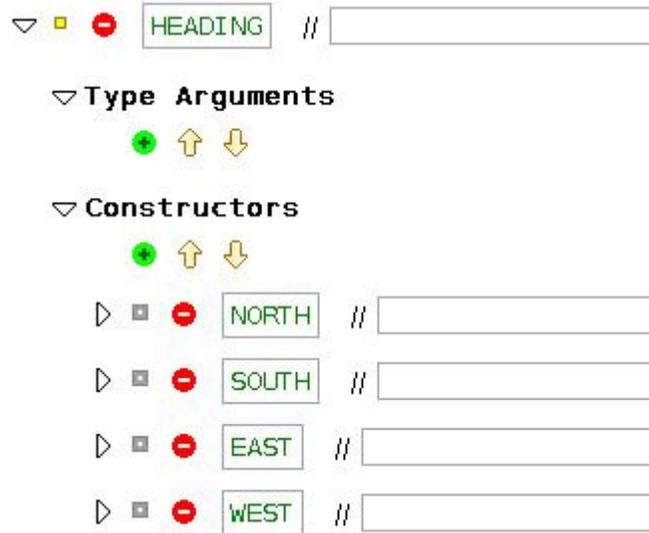


Figure 8. Complete HEADING Definition

Another example of a datatype definition is the List datatype:

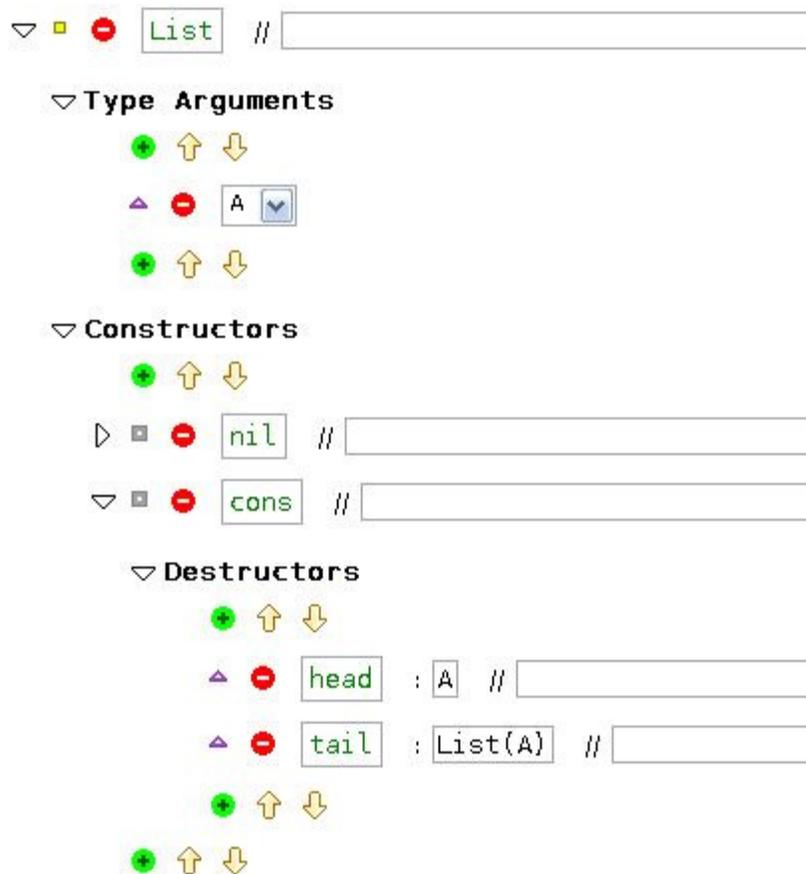


Figure 9. Complete List Definition

In the example above, note that it refers to the type parameter A. This makes the list polymorphic, and as such can be used with any set (e.g., Integer List, BOOL List). Note that, we can also have a List of HEADING (s).

More generally, the following figure describes the components of a datatype definition:

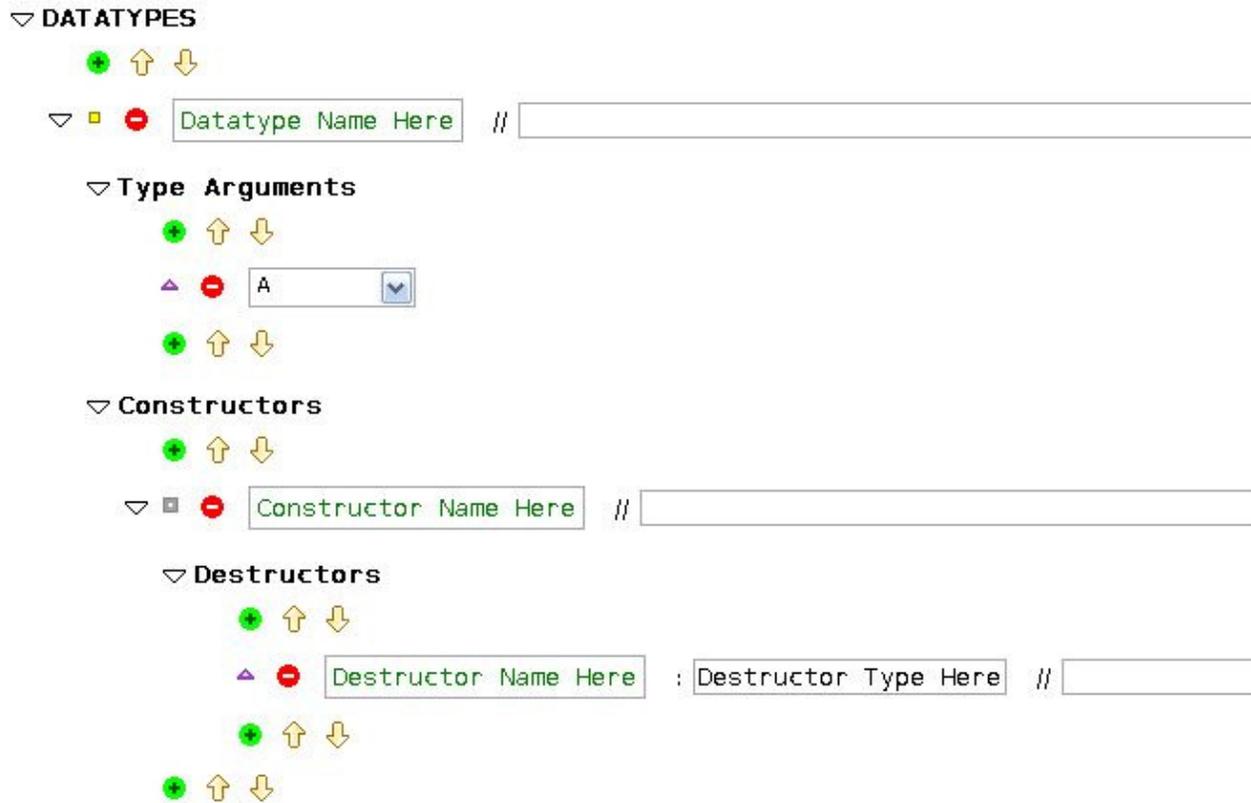


Figure 10. How to Add a Datatype

### Ad Operators

New operators can be defined in the theory by pressing the  button under the Operators heading. The next figure shows two example definitions of an expression operator and a predicate operator.

▼ OPERATORS  
 + ↑ ↓

▼ bagDef : Syntax Bag expression PREFIX Associativity not applicable Commutativity not commutative //

▼ arguments  
 + ↑ ↓  
 ▲ a :  $\mathbb{P}(A)$  //  
 + ↑ ↓

▼ condition  
 + ↑ ↓  
 ✱ finite(a) //  
 + ↑ ↓

▼ direct definition  
 + ↑ ↓  
 ✱  $\{f \cdot f \in a \rightarrow \mathbb{N} \mid f\}$  //  
 + ↑ ↓

▼ isBagDef : Syntax isBag predicate PREFIX Associativity not applicable Commutativity not commutative //

▼ arguments  
 + ↑ ↓  
 ▲ b :  $A \leftrightarrow \mathbb{Z}$  //  
 + ↑ ↓

▼ condition  
 + ↑ ↓

▼ direct definition  
 + ↑ ↓  
 ✱  $\exists a \cdot \text{finite}(a) \wedge b \in \text{Bag}(a)$  //  
 + ↑ ↓

Figure 11. Add new Operators

### Add Theorems

Theorems can be added by pressing + under the theorems heading as follows:

▼ THEOREMS



- ▣ thm0 :  $\forall a. a \in \mathbb{P}(A) \wedge \text{finite}(a) \Rightarrow (\forall b. b \in \text{Bag}(a) \Rightarrow \text{finite}(b))$  //
- ▣ thm1 :  $\forall a. ((a \subseteq A \wedge \text{finite}(a)) \Rightarrow \text{isBag}(\text{emptyBag}(a)))$  //
- ▣ thm2 :  $\forall a, e, b. (a \subseteq A \wedge e \in a \wedge \text{finite}(a) \wedge \text{isBag}(b) \wedge \text{bagContent}(b)=a \Rightarrow \text{isBag}(\text{bagAdd}(e,b)))$  //

Figure 12. Add Theorems

The following figure describes the components of a theorem:

- ▣  :  //

Figure 13. How to Add a Theorem

**Add Proof Rules**

Proof rules can be added by adding one or more rules blocks using under the Rules Block heading:

▼ PROOF RULES



- ▼  : //

▼ Metavariables



- :  $A \leftrightarrow \mathbb{Z}$  //
- :  $\mathbb{P}(A)$  //
- :  $A$  //



▼ Rewrite Rules



▷ Inference Rules



Figure 14. Add a Rules Block

Rules blocks are useful to group together related rules.

### **Add Metavariables**

Metavariables are identifiers that are declared to be used within rewrite and inference rules. Each metavariable has a name and a type. Figure 14 provides three examples of metavariable declarations.

### **Add Rewrite Rules**

Rewrite rules can be added by clicking on  under the Rewrite Rules heading:

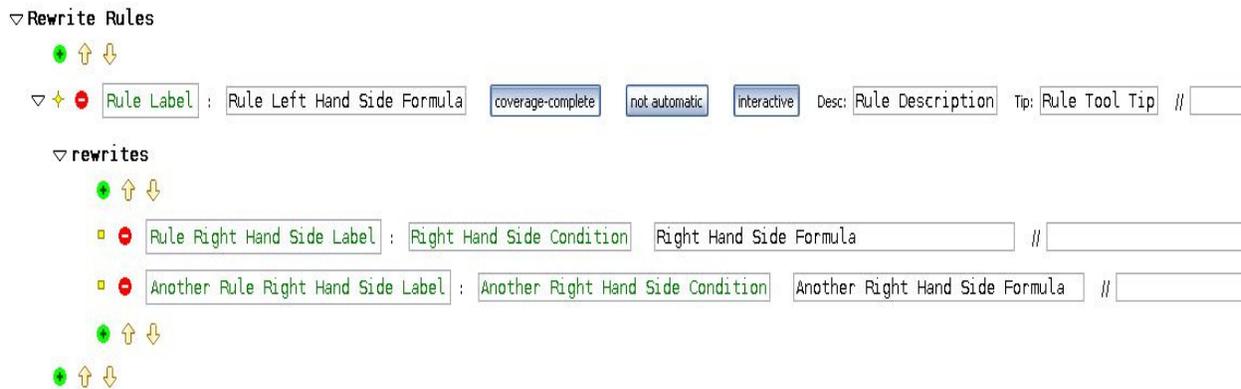


Figure 15. Add a Rewrite Rule

- Coverage completeness indicates whether the rewrite rule covers all cases.
- The right hand side condition is the condition under which the left hand side can be rewritten to the right hand side.
- Rule description is the description that goes into the proof tree (to display) after applying the rule.
- Rule tool tip is the tip that is displayed when a rule is manually applicable.

### **Add Inference Rules**

Inference rules are rules that can be used to infer more hypotheses, split goal into sub-goals and discharge sequents. Inference rules can be added in a similar fashion to rewrite rules by pressing on the appropriate  button.

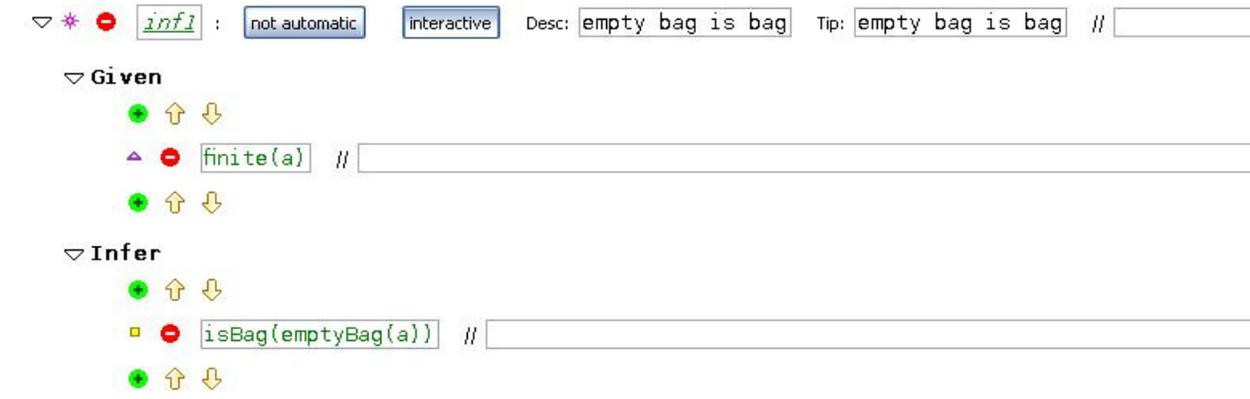


Figure 16. Add an Inference Rule

## Theory Deployment

Theories have associated proof obligations. Currently, these proof obligations are:

- Operator Definitions :
  - o /Op-WD → Operator Well-Definedness Preservation
  - o /Op-COMMUT → Operator Commutativity;
  - o /Op-ASSOC → Operator Associativity
- Rewrite Rules :
  - o /WD-S/ → Rule RHS Well-Definedness Preservation
  - o /WD-C/ → Rule RHS Condition Well-Definedness Preservation
  - o /S/ → Rule RHS Soundness (equality or equivalence)
  - o /C → Rule Completeness

Once the user is satisfied with the theory, it can then be deployed to be used by models and other theories under development.

### Workspace Scope of Theories

Theories are created within and deployed to the same project, “**MathExtensions**” project. The following steps describe how to deploy a theory:

- Open the theory file that you want to deploy using the Event-B Theory Editor.
- Click on the button. The wizard in Figure 18 will show up:

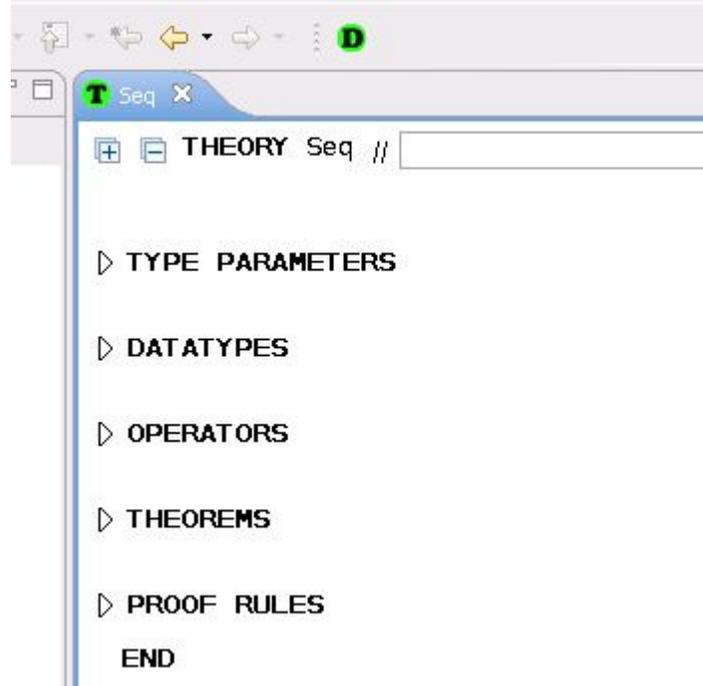


Figure 17. Deploy from Editor View

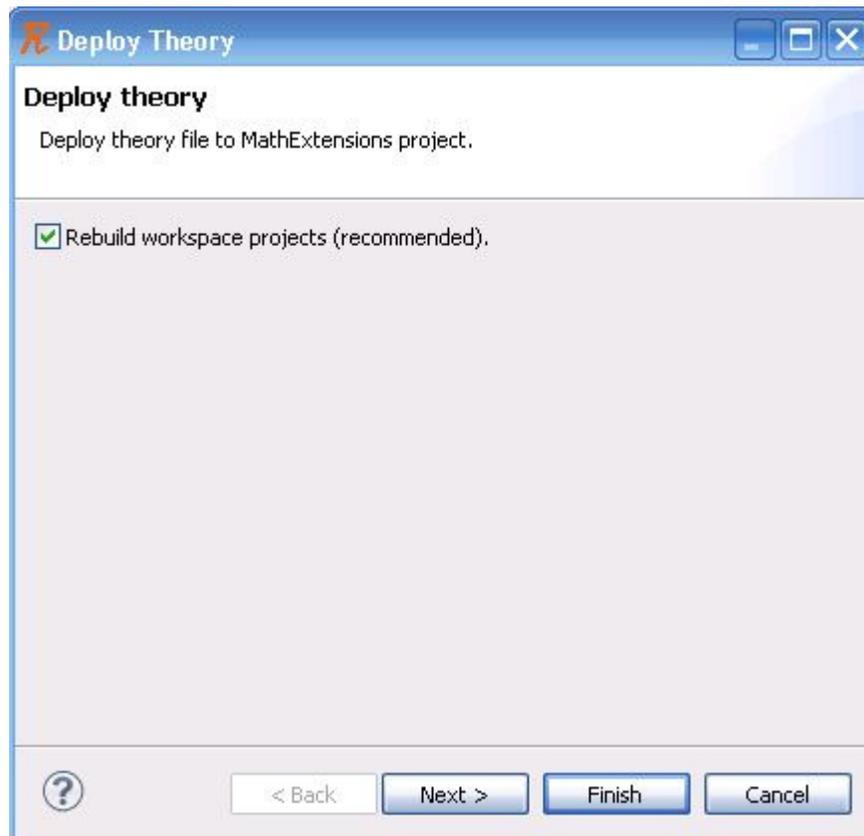


Figure 18. Deployment Dialog

- If a workspace rebuild is desired, keep the radio button checked.
- Press Finish. Alternatively, to see some information about the theory, click Next. The following dialog will show up:

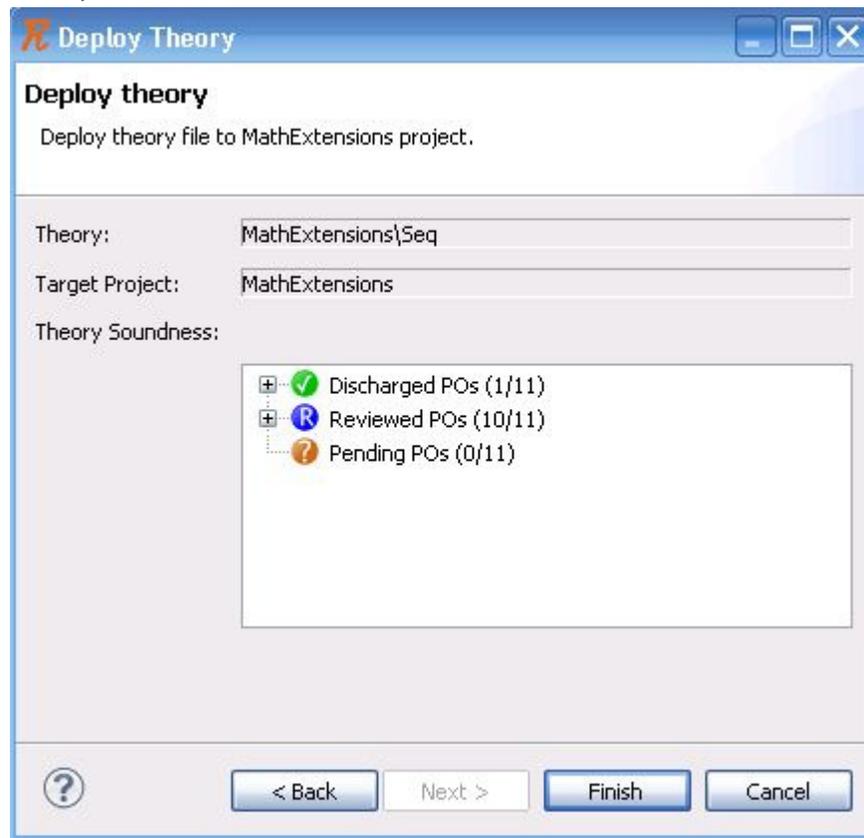


Figure 19. Theory Information

The theory should now be available to be used in models and other theories under development.