CamilleX Documentation

None

Thai Son Hoang

None

Table of contents

1. CamilleX User Manual

CamilleX new constructs (called `XMachines` and `XContexts`) for `Event-B` modelling. The new constructs are text files which are automatically translated into the corresponding `Rodin`'s `Event-B` constructs (i.e., `Machines` and `Contexts`) accordingly. Facility for translating to and from `Rodin`'s components to `CamilleX` components can be invoked manually. `CamilleX` is inspired by Camille text editor for Rodin and is based on XText technology, hence the name `CamilleX`.

- *Getting Started*:
- *Installation*: Information for installing the *CamilleX* feature.
- *Basic tutorial*: This tutorial provides a step-by-step walk-through working with CamilleX constructs.

2. Getting Started

2.1 Installation

CamilleX is available from the main Rodin update site (under `CamilleX` category). There are two versions of the feature, the standard version for users and the SDK version for software developers which include source code.

2.2 Configuration

Windows users must change the workspace text file encoding to *UTF-8*. This can be updated under the `Rodin Preferences` `General/Workspace` then in the `Text file encoding` section, select Other: `UTF-8`.

2.3 IMPORTANT

Currently, *CamilleX* not only supports *standard* Event-B machines and contexts, but also supports *Machine Inclusion* (for composition), and *Record* extension to the Event-B modelling language.

Since the *XContexts* and *XMachines* are compiled to the Rodin files, the corresponding Rodin contexts and machines will be **OVER-WRITTEN**. Any changes in the Rodin files will not be lost.

DO NOT USE the *CamilleX* if you use modelling plug-ins that use the Rodin files as source such as *UML-B* state-machines and class-diagrams, as the additional modelling elements will be over-written.

2.4 Basic Tutorial

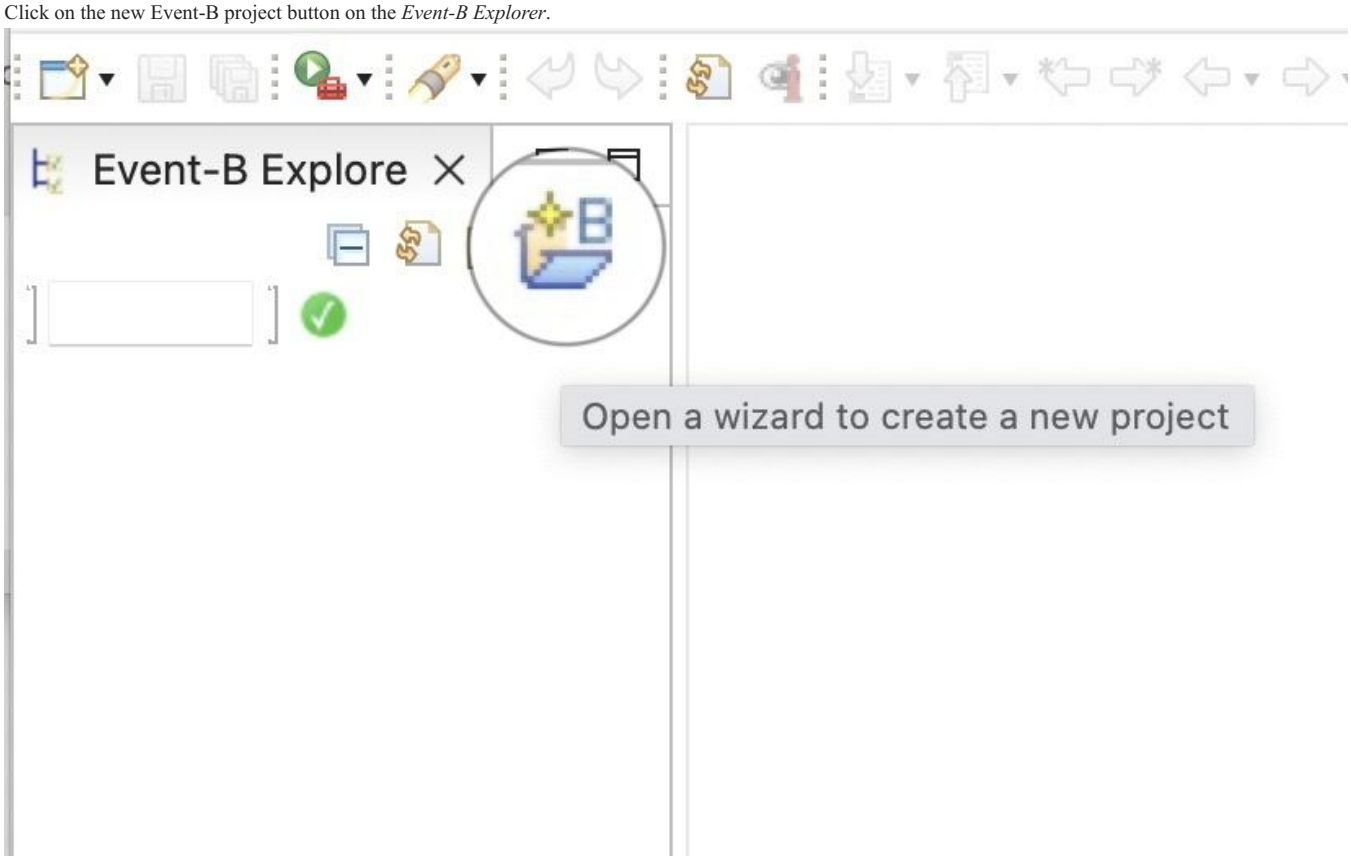
2.4.1 Task 1. Create an Event-B Project

Introduction

The purpose of this task is to create an Event-B project for the CamilleX constructs.

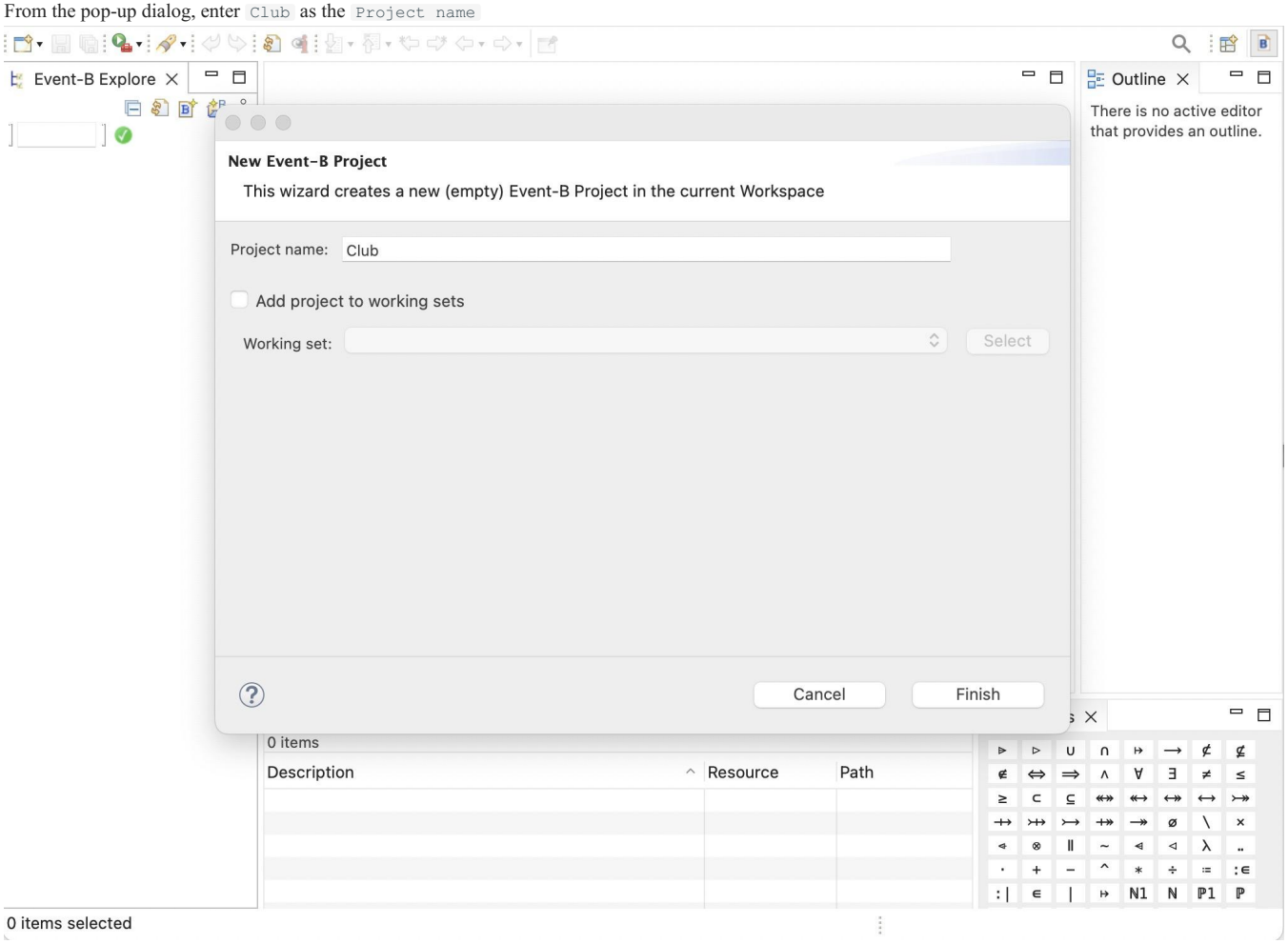
Step 1. Create a New Event-B Project Named `Club`

• Click on the new Event-B project button on the *Event-B Explorer*.

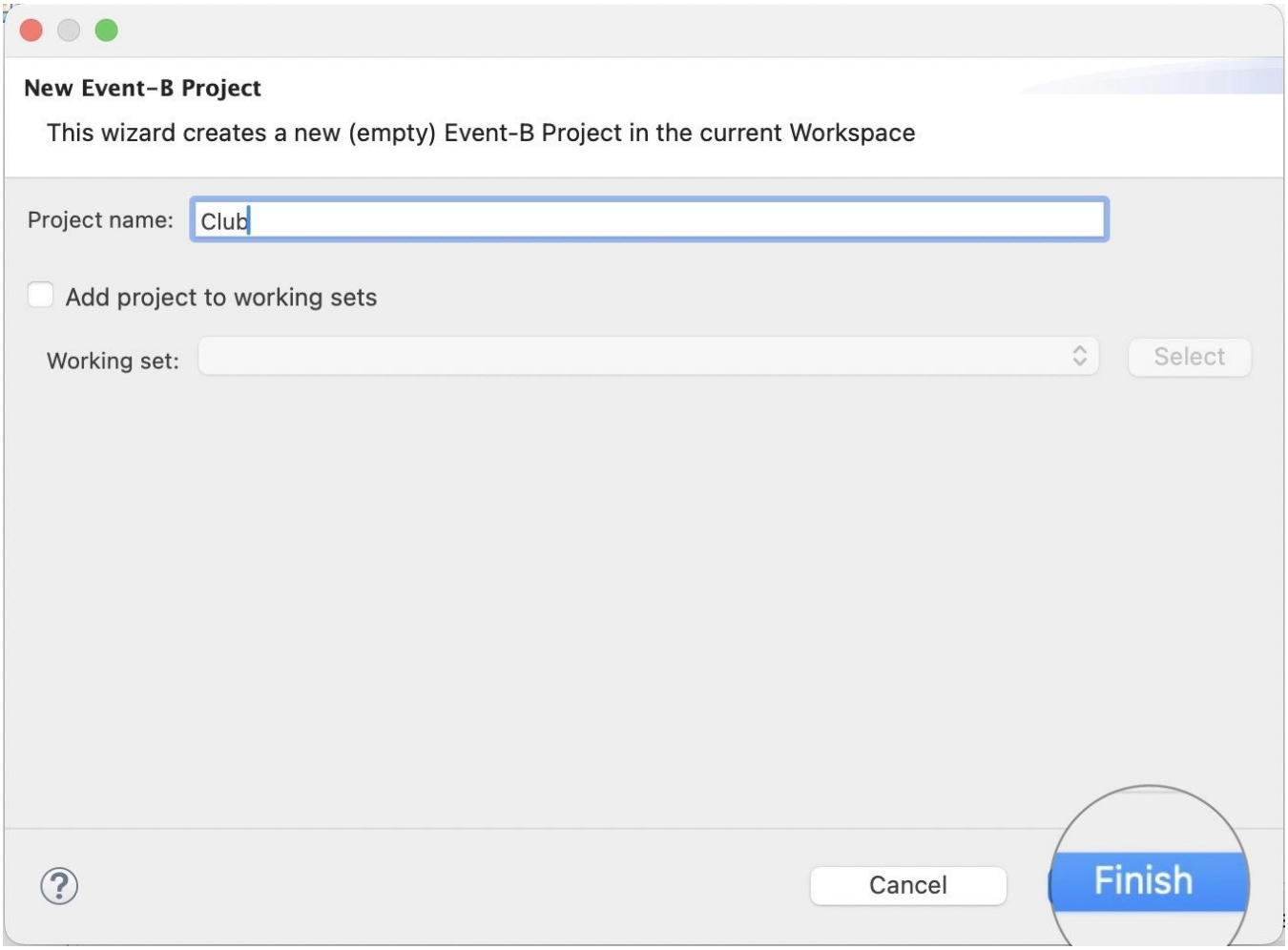


(The same wizard can be invoke through the menu `File -> New -> Event-B Project`)

• From the pop-up dialog, enter `Club` as the `Project name`

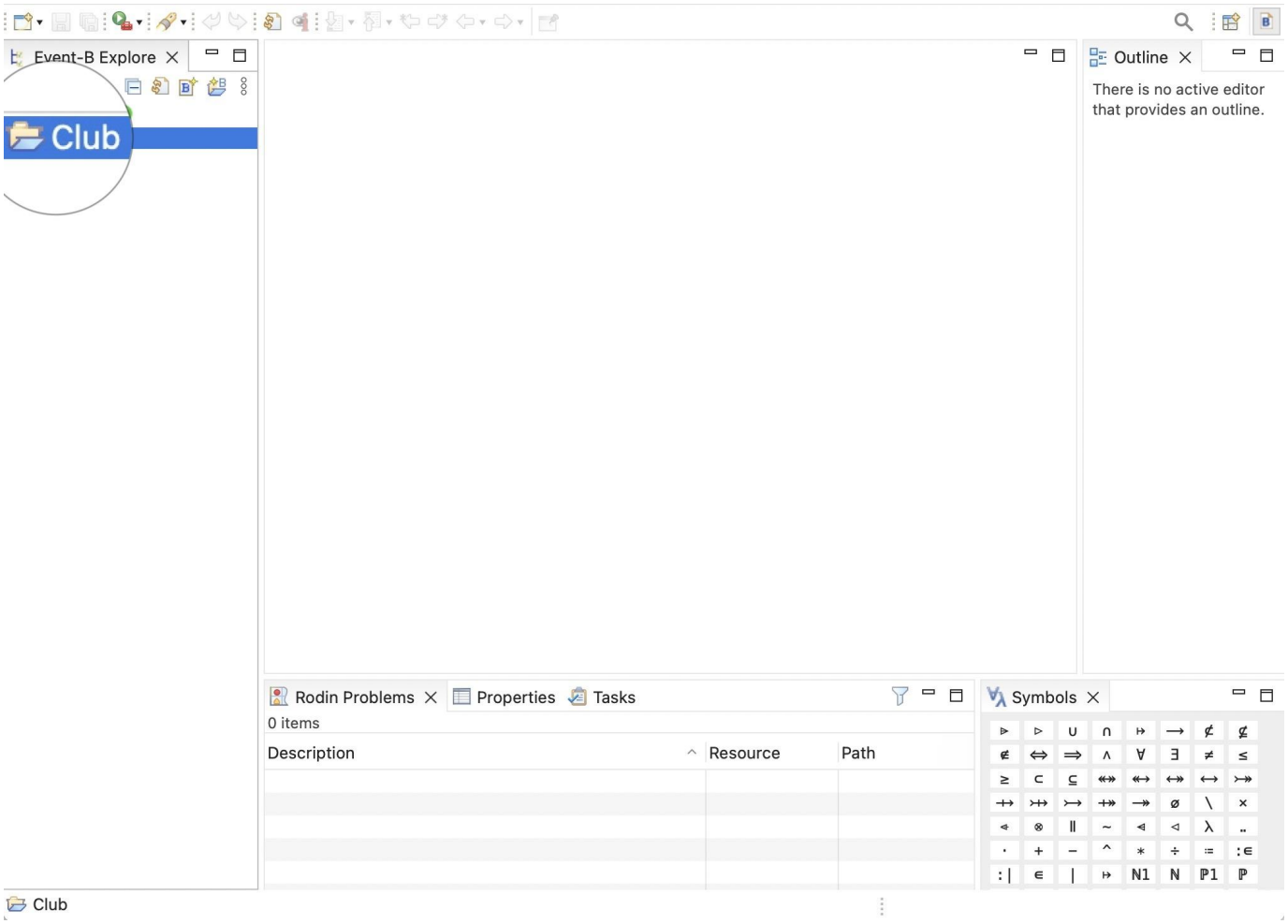


• Click `Finish` to confirm the creation of the project.



Conclusion

By now, the project `Club` should be visible in the *Event-B Explorer*.



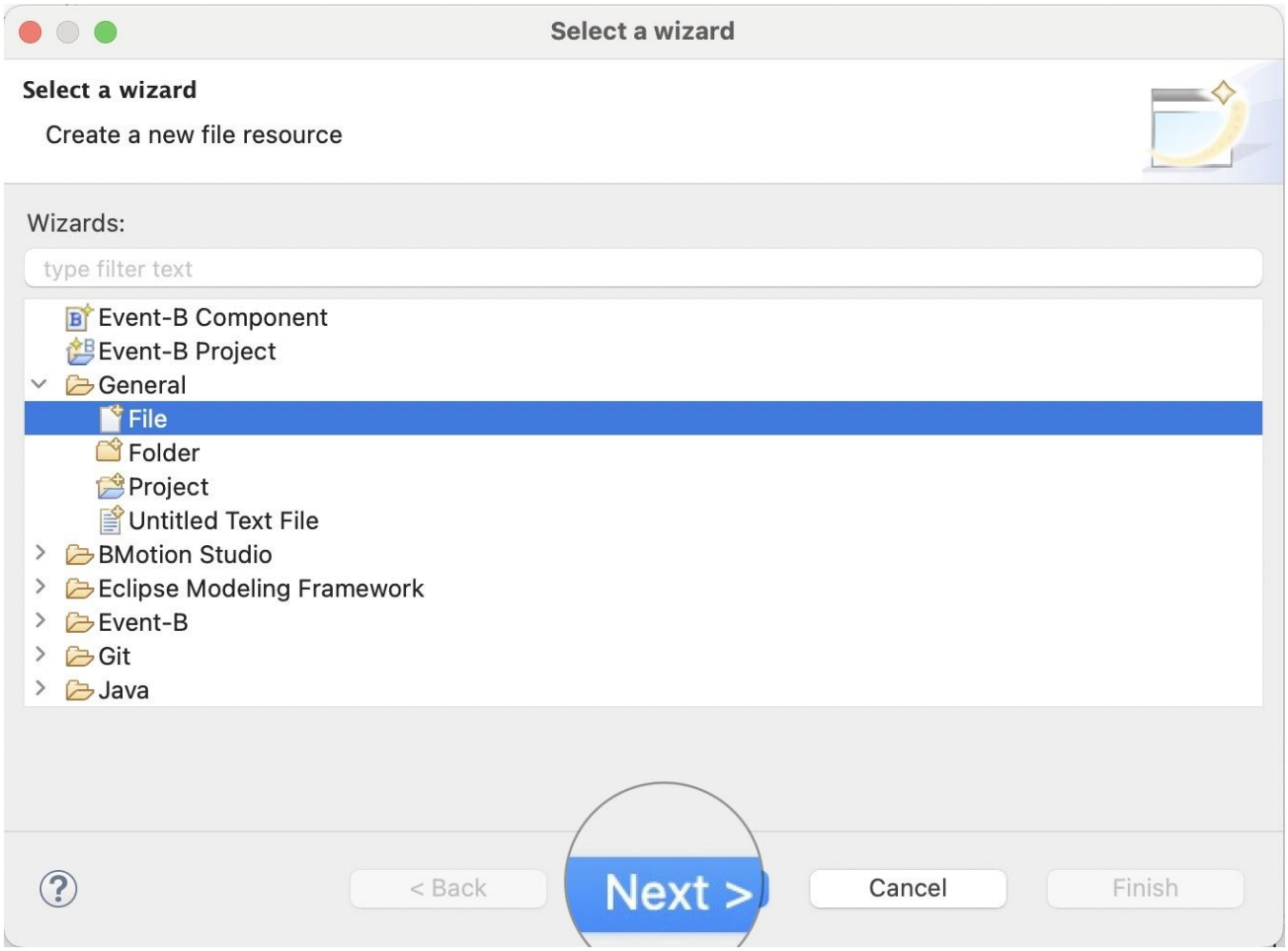
2.4.2 Task 2. Create an XContext

Introduction

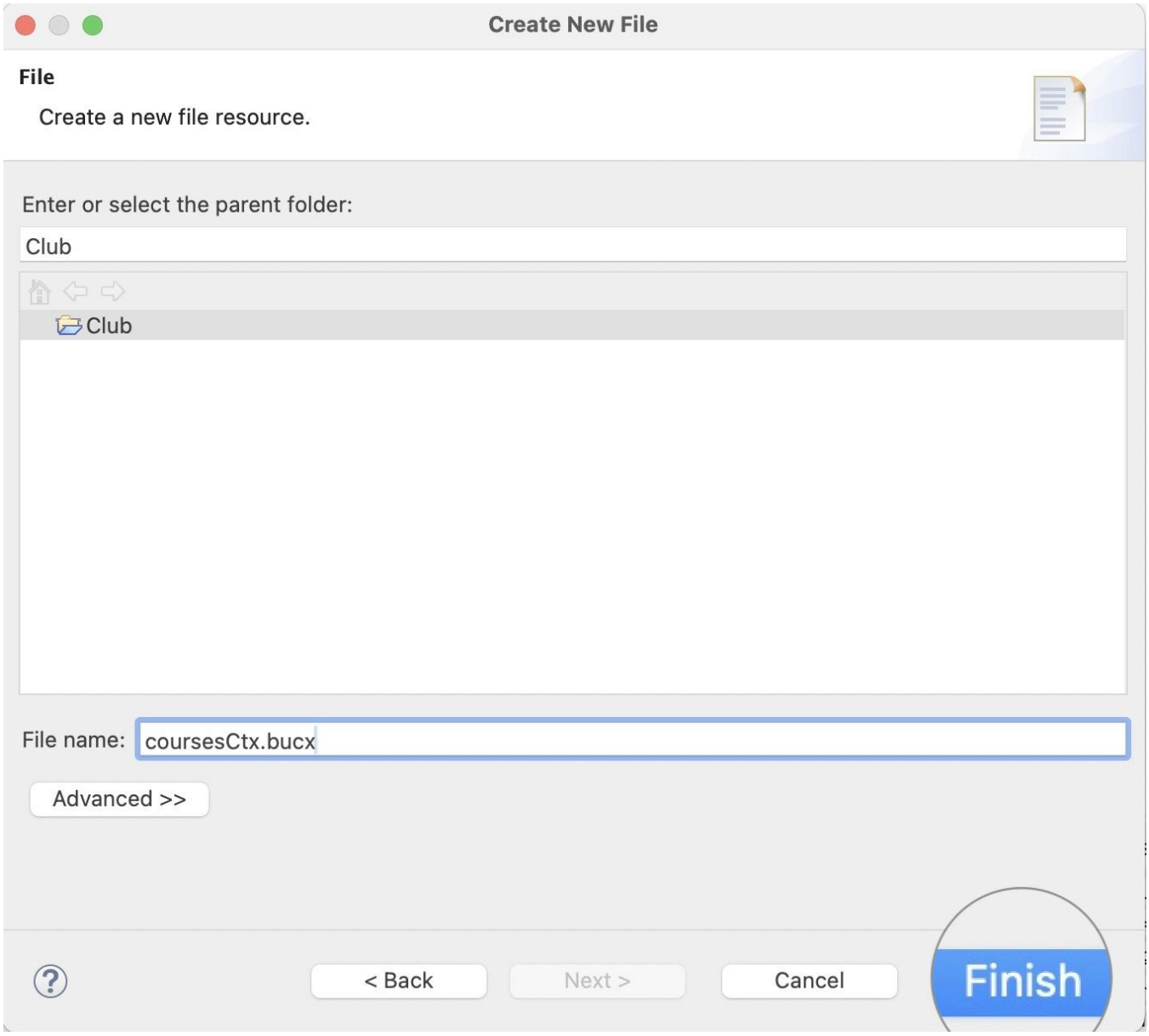
The purpose of this task is to create a simple XContext within the newly created project.

Step 1. Create a New XContext Named `coursesCtx.bucx`

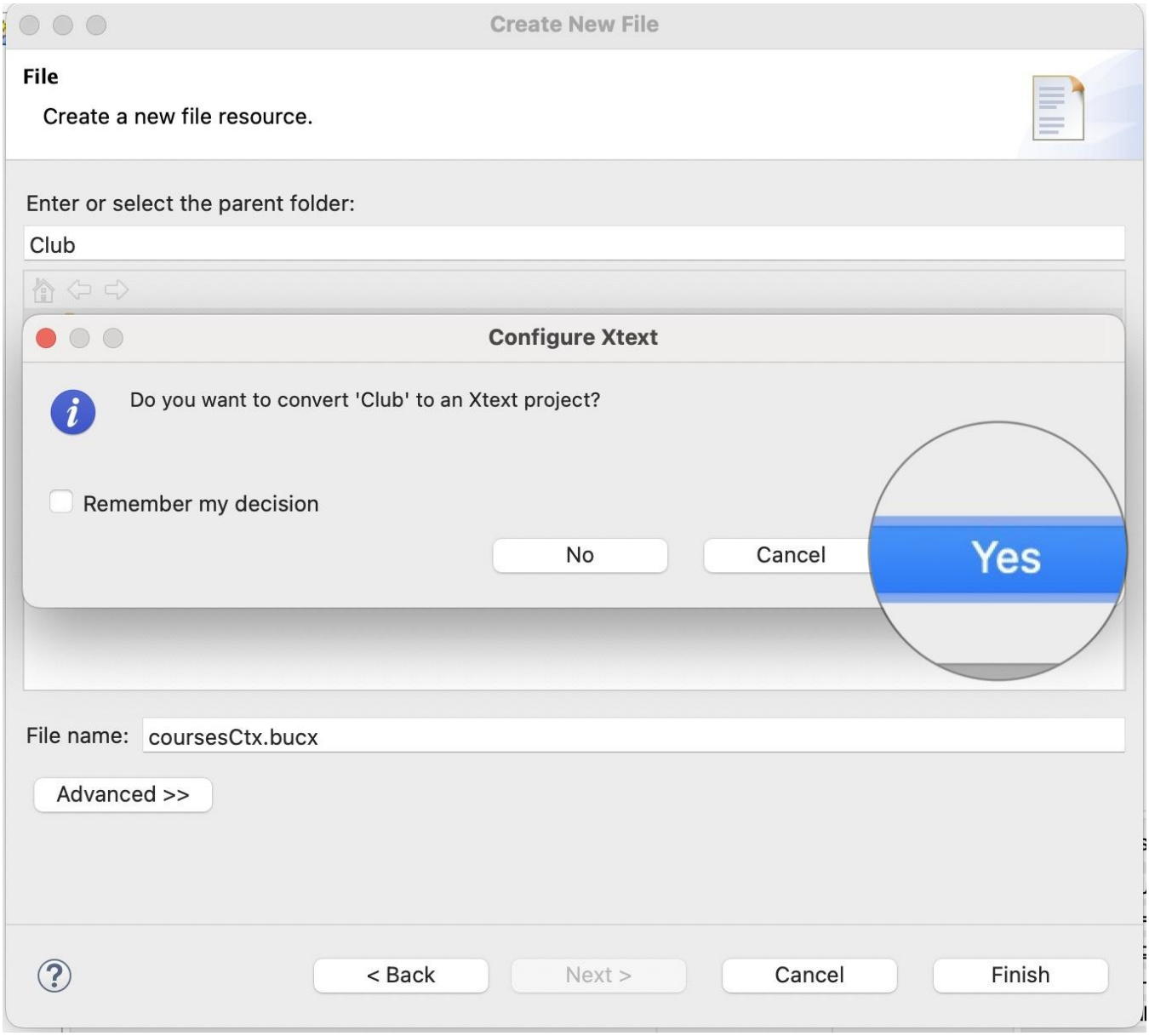
- Use the menu `File -> New -> Other` to open the `Select a wizard` dialog.
- On the pop-up `Select a wizard` dialog, navigate to `General -> File`, click `Next`.



• On the `Create New File` dialog, choose `Club` project as the parent folder, and put `coursesCtx.bucx` as the `File name`. The file extension `.bucx` is important to indicate that the file is an *XContext*. Click `Finish` to confirm the file creation.



• **Important**: A pop-up dialog will be displayed asking to convert the `Club` project to an *XText* project, please answer **Yes**. This enables the *XText* builder to work automatically for converting CamilleX constructs to Rodin constructs.



(If you miss this step, you can invoke it via right click on the `Club` project from the *Event-B Explorer* and `Configure -> Convert to XText Project`). The new created file `coursesCtx.bucx` will be opened automatically in an editor. It has some error markers and we will fix this in the next step.

Step 2. Set the Content of `courseCtx.bucx`

- Using the editor, set the content of `coursesCtx.bucx` as follows.

```
context coursesCtx
sets
    CRS     // The set of all courses
constants
    m       // The maximum number of courses
axioms
    @axm0_1: finite(CRS)    // There can only be a finite number of courses
    @axm0_2: m ∈ ℕ1         // The maximum number of courses is a non-zero natural number
theorem @thm0_1: 0 < m      // The maximum number of courses is positive
end
```

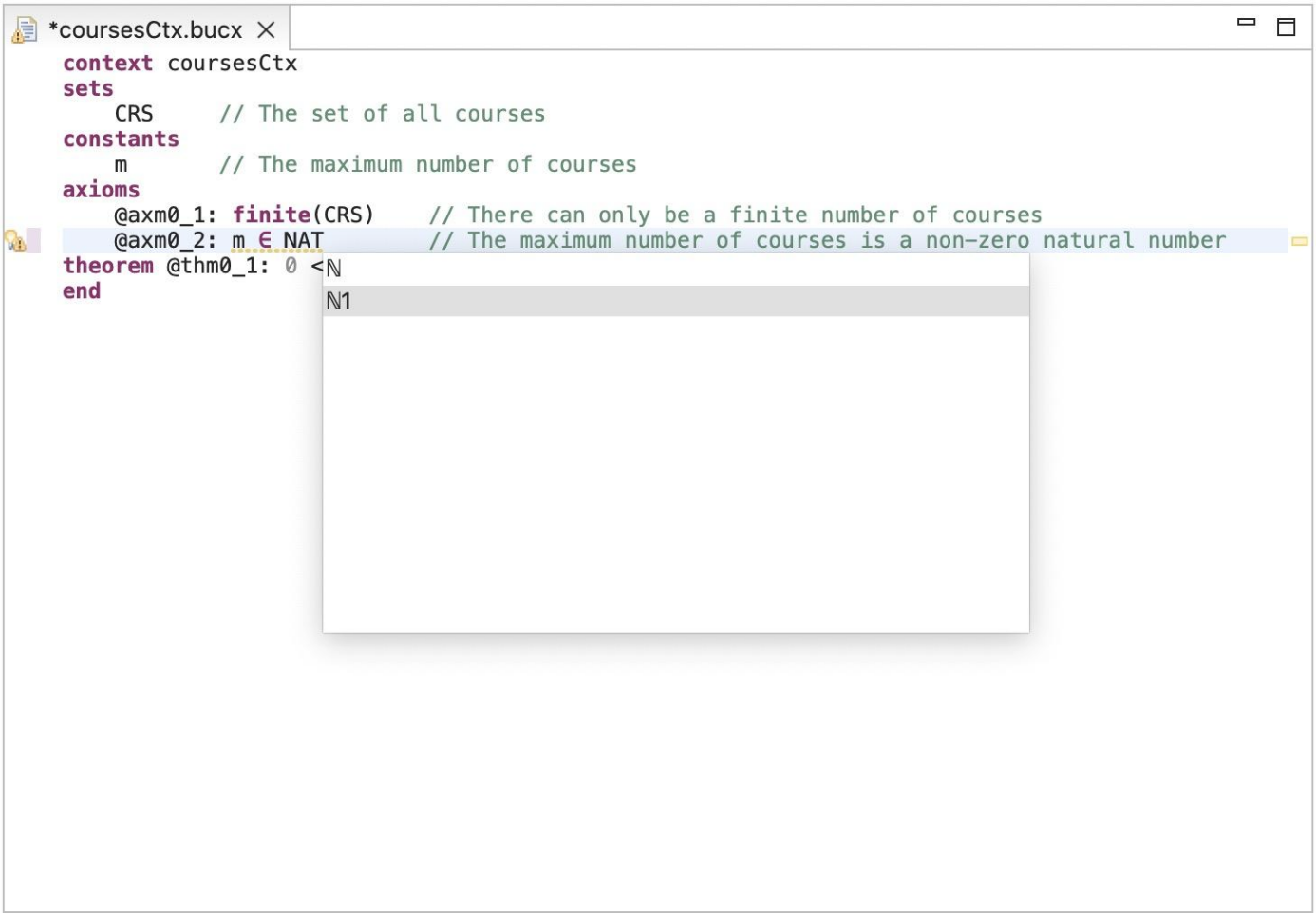
coursesCtx.bucx

```
context coursesCtx
sets
    CRS // The set of all courses
constants
    m // The maximum number of courses
axioms
    @axm0_1: finite(CRS)    // There can only be a finite number of courses
    @axm0_2: m ∈ ℕ1         // The maximum number of courses is a non-zero natural number
theorem @thm0_1: 0 < m      // The maximum number of courses is positive

axiom
    @axm0_3: m ≤ card(CRS)
end
```

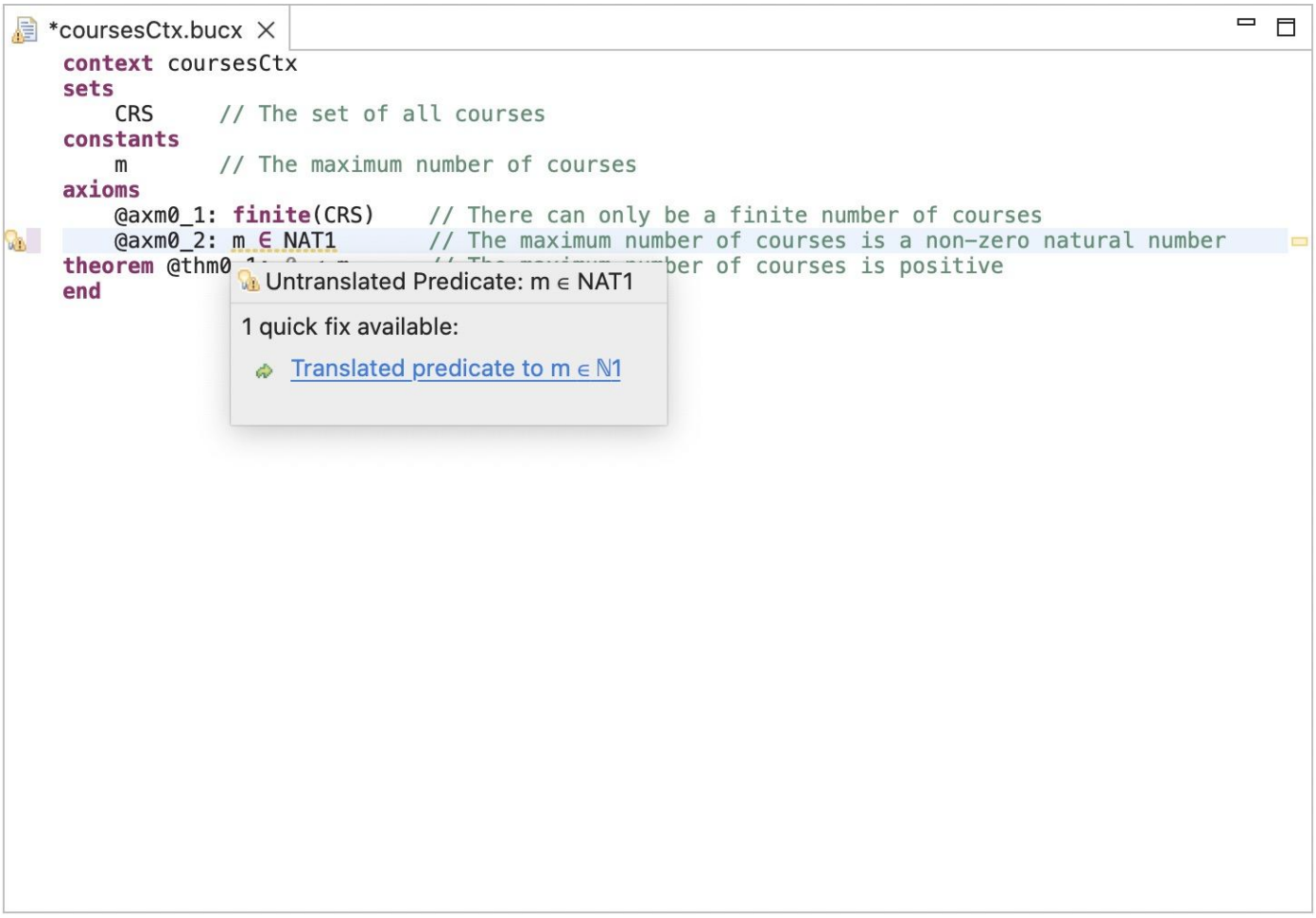
TYPESETTING MATHEMATICAL SYMBOLS

In order to typeset Event-B mathematical symbols, e.g., `N1`, there are three different approaches.

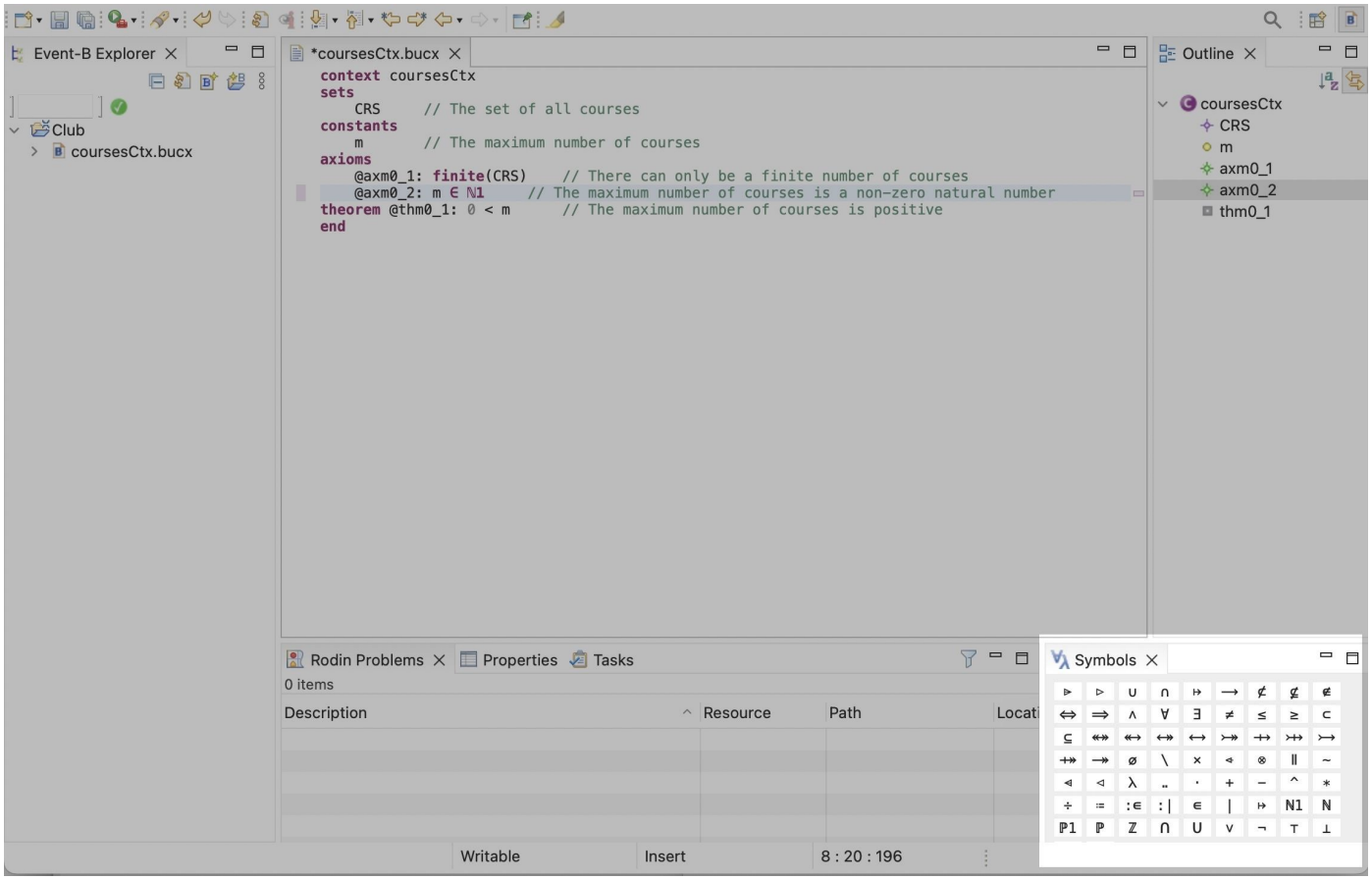
1. Using *Content Assist*. *Content Assist* can translate *ASCII* characters into Unicode symbols. For example, when typing `NAT` and invoking content assist (e.g., on `Ctrl + Space` on Mac OS), a dropdown list will appear with options for typesetting `N` and `N1`.



2. Using *Quick Fix*. The *CamilleX* editor offer quick fixes for ASCII untranslated formula. Untranslated formula are indicated by warnings with yellow squiggly lines under the formula. Hover the mouse over the untranslated formulae, a pop-up dialog will appear to offer to translate the formulae.



3. Using *Symbols* Table. Symbols can be inserted into the *CamilleX* editor. (If the *Symbols* table is not visible in your Rodin, you can open it from the menu `Window -> Show View -> Symbols`.

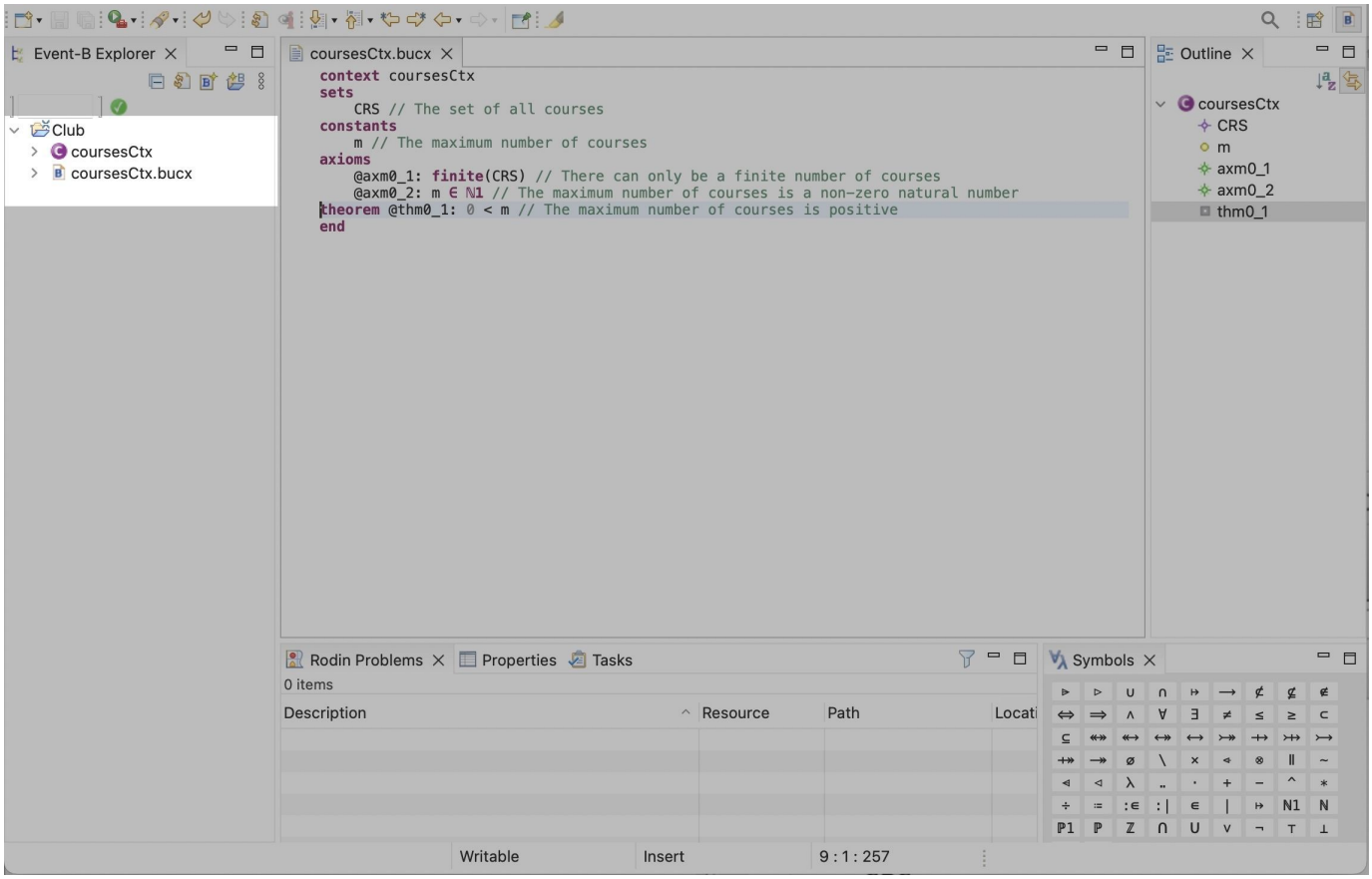


Step 3. Save the `coursesCtx.bucx` file

Save the file `coursesCtx.bucx`, the *XText* builder will generate Rodin context `coursesCtx` automatically.

Conclusion

By now, the XContext `coursesCtx.bucx` and the corresponding Rodin Context `coursesCtx` should be visible in the Event-B Explorer.



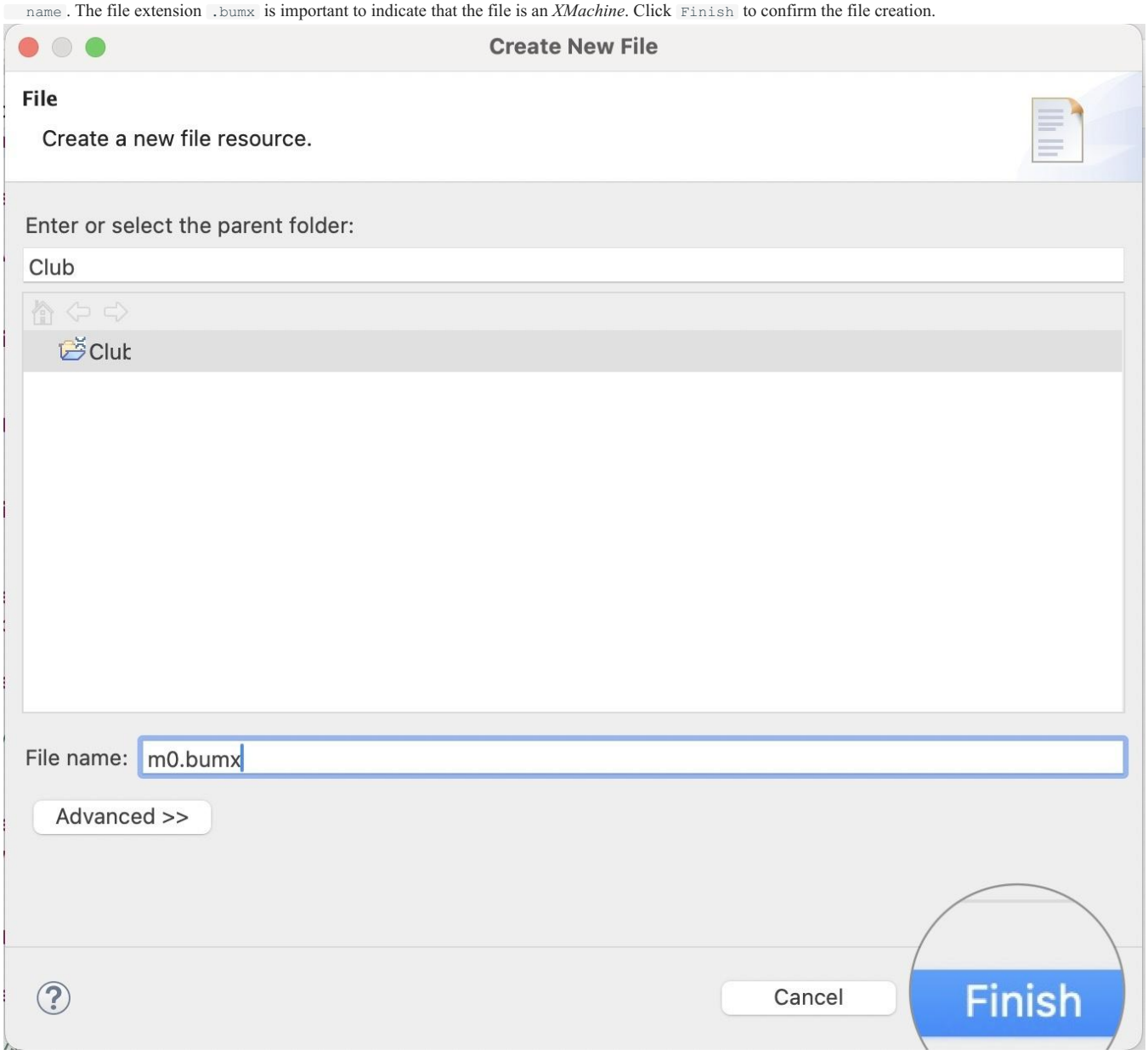
2.4.3 Task 3. Create an XMachine

Introduction

The purpose of this task is to create a simple XMachine within the newly created project.

Step 1. Create a New XMachine Named `m0.bumx`

- Right click on the `Club` project and then `New -> File` to open the `Create New File` dialog. On the `Create New File` dialog, choose `Club` project as the parent folder, and put `m0.bumx` as the `File name`. The file extension `.bumx` is important to indicate that the file is an *XMachine*. Click `Finish` to confirm the file creation.



Step 2. Set the Content of `m0.bumx`

- Using the editor, set the content of `m0.bumx` as follows.

```
machine m0

sees coursesCtx

variables
```

```
    crs     // The set of existing courses

invariants
    @inv0_1: crs ⊆ CRS

theorem
    @thm0_2: finite(crs)

invariant
    @inv0_2: card(crs) ≤ m

event INITIALISATION
begin
    @act1: crs ≔ ∅
end

/*
 * Event to open a set of courses using non-deterministic assignment.
 */
event OpenCourses
when
    @grd0_1 : card(crs) ≠ m
    theorem @thm0_3 : crs ≠ CRS
then
    @act0_1 : crs :| crs ⊂ crs' ∧ card(crs') ≤ m
end

/*
 * Event to close a set of courses using event parameters
 */
anticipated event CloseCourses
any cs
where
    @grd1: cs ⊆ crs
    @grd2: cs ≠ ∅
then
    @act1: crs ≔ crs ∖ cs
end
```

m0.bumx

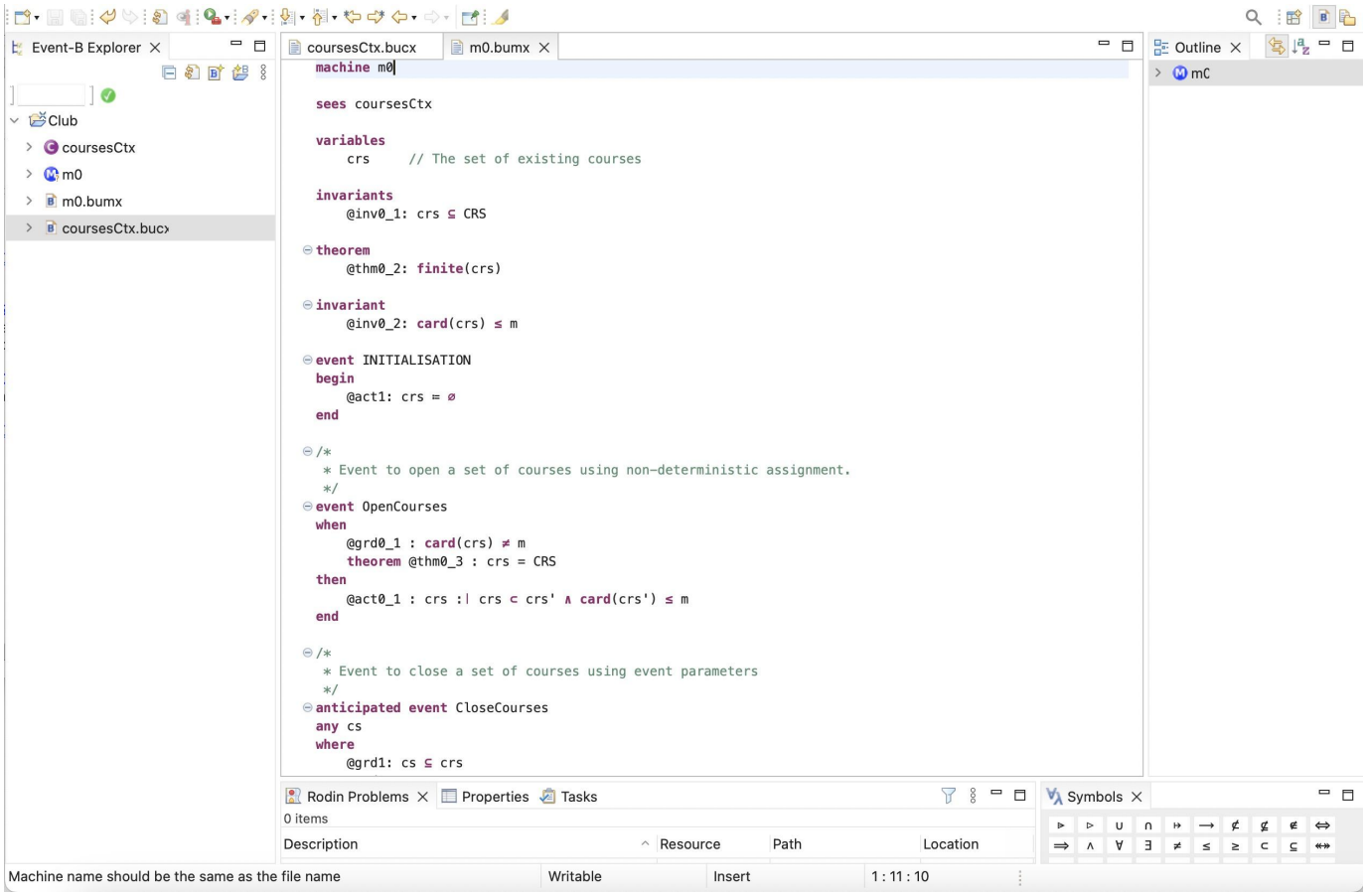
```
machine m0

sees coursesCtx

variables
    crs     // The set of existing courses

invariants
    @inv0_1: crs ⊆ CRS

theorem
    @thm0_2: finite(crs)

invariant
    @inv0_2: card(crs) ≤ m

event INITIALISATION
begin
    @act1: crs ≔ ∅
end

/*
 * Event to open a set of courses using non-deterministic assignment.
 */
event OpenCourses
when
    @grd0_1 : card(crs) ≠ m
    theorem @thm0_3 : crs ≠ CRS
then
    @act0_1 : crs :| crs ⊂ crs' ∧ card(crs') ≤ m
end

/*
 * Event to close a set of courses using event parameters
 */
anticipated event CloseCourses
any cs
where
    @grd1: cs ⊆ crs
```

Step 3. Save the `m0.bumx` file

Save the file `m0.bumx`, the *XText* builder will generate Rodin machine `m0` automatically.

Conclusion

By now, the XMachne `m0.bucx` and the corresponding Rodin Machine `m0` should be visible in the Event-B Explorer.



2.4.4 Task 4. Create an extended XContext

Introduction

The purpose of this task is to create an extended XContext within the newly created project.

Step 1. Create a simple context XContext Named `membersCtx.bucx`

- Follow the steps in Task 2 to create a context named `membersCtx.bucx` with the following content.

```
context membersCtx

sets MEM

axioms
    @axm1_1: finite(MEM)

end
```

Step 2. Create an extended context XContext Named `participantsCtx.bucx`

- Follow the steps in Task 2 to create a context named `participantsCtx.bucx` with the following content.

```
context participantsCtx

extends membersCtx

constants
    PRTCPT

axioms
    @axm1_2: PRTCPT ⊆ MEM

theorem
    @thm1_1: finite(PRTCPT)
end
```

The `extends` clause signifies that this context extends `membersCtx`.

Step 3. Create an extended context XContext Named `instructorsCtx.bucx`

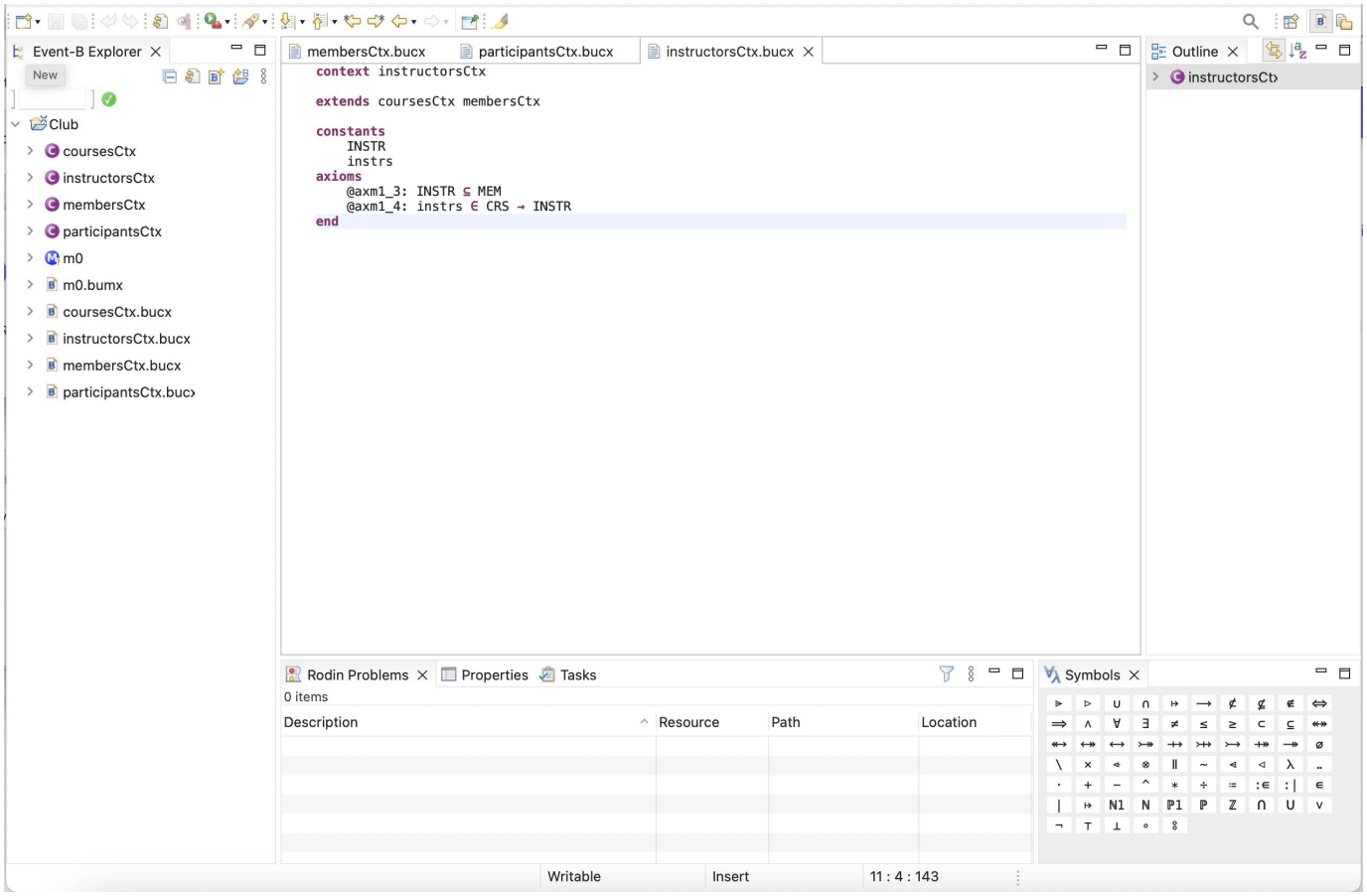
- Follow the steps in Task 2 to create a context named `instructorsCtx.bucx` with the following content.

```
context instructorsCtx

extends coursesCtx membersCtx

constants
    INSTR
    instrs
axioms
    @axm1_3: INSTR ⊆ MEM
    @axm1_4: instrs ∈ CRS → INSTR
end
```

The `extends` clause signifies that this context extends both `coursesCtx` and `membersCtx`.

Conclusion

By now, the XContexts `membersCtx.bucx`, `participantsCtx.bucx`, `instructorsCtx` and their corresponding Rodin Contexts should be visible in the Event-B Explorer.



2.4.5 Task 4. Create a refined XMachine

Introduction

The purpose of this task is to create a refined XMachine within the newly created project. We will use the abstract Rodin Machine as the source to create a template for a refined XMachine. We explore the translation from Rodin constructs to CamilleX constructs.

Step 1. Create a the Rodin refined machine `m1.bumx`

We will use the *Refine* wizard to create the Rodin refined machine. - Right-click on the Rodin `m0` file on the *Event-B Explorer*, and choose `Refine` from the context menu. Refine Wizard - On the `New REFINES clause` dialog, put `m1` as the name of the new machine. (`m1` should be the name by default). Click `OK` to confirm the creation of the refined machine `m1`. Create a New M1 Machine

- A new machine `m1` is created and opened automatically. Rodin m1 in the File Explorer

Step 2. Create a refined XMachine Named `m1.bumx`

We will now use the tool to convert the Rodin machine `m1` to the corresponding CamilleX XMachine `m1.bumx`. - Right-click on the Rodin `m1` file on the *Event-B Explorer*, and choose `Convert to CamilleX` from the context menu. Convert to CamilleX Context Menu - The tool will produce the XMachine `m1.bumx`. Double click on the file from the *Event-B Explorer* to open the file in the CamilleX editor. The content of the template `m1.bumx` is as follows. The Template CamilleX m1.bumx

Step 2. Set the content for `m1.bumx`

Use the editor to change the content of `m1.bumx` as follows. - `m1.bumx` sees contexts `participantsCtx` and `instructorsCtx`.

```
sees instructorsCtx participantsCtx
```

- `m1.bumx` has a new variable `prtcpts` (in addition to the current variable `crs`.

```
variables
    crs
    prtcpts
```

- `m1.bumx` has new invariants

```
invariants
    @inv1_1 : prtcpts ∈ crs ↔ PRTCPT
    @inv1_2 : ∀c · c ∈ crs ⇒ instrs(c) ∉ prtcpts[{c}]
```

- The initialisation `INITIALISATION` has a new action. (Notice that it also extends the abstract `INITIALISATION`)

```
event INITIALISATION extends INITIALISATION
begin
    @act2: prtcpts ≔ ∅
end
```

- The `CloseCourses` event has a new action.

```
anticipated event CloseCourses extends CloseCourses
begin
    @act2: prtcpts ≔ cs ⩤ prtcpts
end
```

- Add a new `Register` event

```
convergent event Register
any p c where
    @grd1: p ∈ PRTCPT
    @grd2: c ∈ crs
    @grd3: p ≠ instrs(c)
    @grd4: c ↦ p ∉ prtcpts
then
```

```
    @act1: prtcpts ≔ prtcpts ∪ {c ↦ p}
end
```

- `m1.bumx` has a variant to prove the convergence of event `Register`.

```
variant
    @var1: (crs × PRTCPT) ∖ prtcpts
```

Conclusion

By now, the XMachine `m1.bumx` and the corresponding Rodin Machines should be visible in the Event-B Explorer.