Context instantiation plug-in: a new approach to genericity in Rodin*

Guillaume Verdier[1], Laurent Voisin[2]

[1] LMF/CentraleSupélec, Université Paris-Saclay
[2] Systerel

June 8th, 2021
9th Rodin User and Developer Workshop

*This work is supported by the French ANR project Event-B Rodin Plus (EBRP, ANR-19-CE25-0010).

Introduction

- the core Rodin platform has many features, but lacks genericity/type parametricity
- how to define, prove and use generic mathematical theories or data structures?
  - implement specific instances (e.g., List_Int) with copy-and-paste... which is cumbersome and error-prone
  - use the Theory plug-in, which introduces a new type of files, a slightly different syntax, a deployment process, ...
- as part of the EBRP project, Jean-Raymond Abrial suggested a new, lighter approach: the *context instantiation plug-in* experiments it

Overview

Main idea: start from what people do by hand ("copy-and-paste method") and provide a plug-in to automate it in a safe way.

- reuse contexts
- use carrier sets and constants as generic parameters
  - carrier sets: type parameters
  - constants: value/expression parameters
- prove theorems about these abstract types and constants
- in other contexts, *instantiate* these generic theorems by replacing carrier sets and constants with concrete types and values

Generic contexts

"Generic" contexts are written as usual and do not need the plug-in. For example:

context sequence
sets S
constants
  seq
  add
  size
axioms
  axm1: $seq = \{f \mapsto n | n \in \mathbb{N} \land f \in 1 \mathinner{..} n \to S\}$
  axm2: $add = (\lambda x \cdot x \in S | (\lambda f \mapsto n \cdot f \mapsto n \in seq | f \cup \{n+1 \mapsto x\} \mapsto n+1))$
  theorem thm1: $add \in S \to (seq \to seq)$
  axm3: $size = (\lambda f \mapsto n \cdot f \mapsto n \in seq | n)$
  theorem thm2: $size \in seq \to \mathbb{N}$
  theorem thm3: $\forall x, f, n \cdot x \in S \land (f \mapsto n \in seq) \Rightarrow size(add(x)(f \mapsto n)) = size(f \mapsto n) + 1$

Instantiations

In the most basic setting, users can manually instantiate theorems and describe the instantiation in the comment box: the plug-in checks that the instantiation is valid.
Useful to check existing contexts:

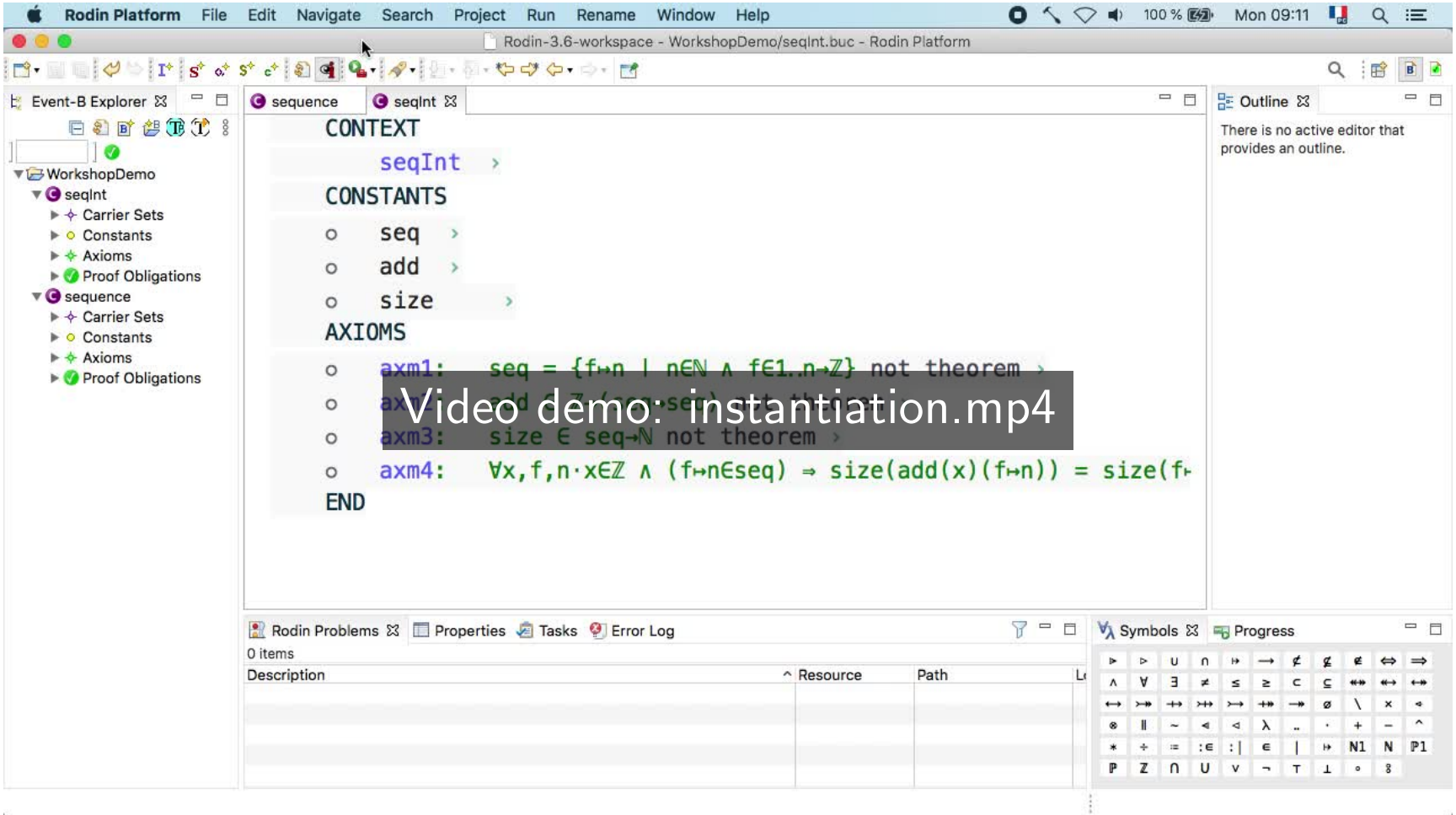
context seqInt
constants
seq
add
size
axioms
@axm1: $seq = \{f \mapsto n | n \in \mathbb{N} \land f \in 1 \mathinner{..} n \rightarrow \mathbb{Z}\}$ // *sequence*|*S*:=$\mathbb{Z}$|*axm1*
@axm2: $add \in \mathbb{Z} \rightarrow (seq \rightarrow seq)$ // *sequence*|*S*:=$\mathbb{Z}$|*thm1*
@axm3: $size \in seq \rightarrow \mathbb{N}$ // *sequence*|*S*:=$\mathbb{Z}$|*thm2*
@axm4: $\forall x, f, n \cdot x \in \mathbb{Z} \land f \mapsto n \in seq \Rightarrow size(add(x)(f \mapsto n)) = size(f \mapsto n) + 1$ // *sequence*|*S*:=$\mathbb{Z}$|*thm3*

Syntax:

- @ before axiom label
- *context* | *substitutions* | *theorem* in comment
  - substitutions: $param_1 := value_1$;; $param_2 := value_2$;; ...

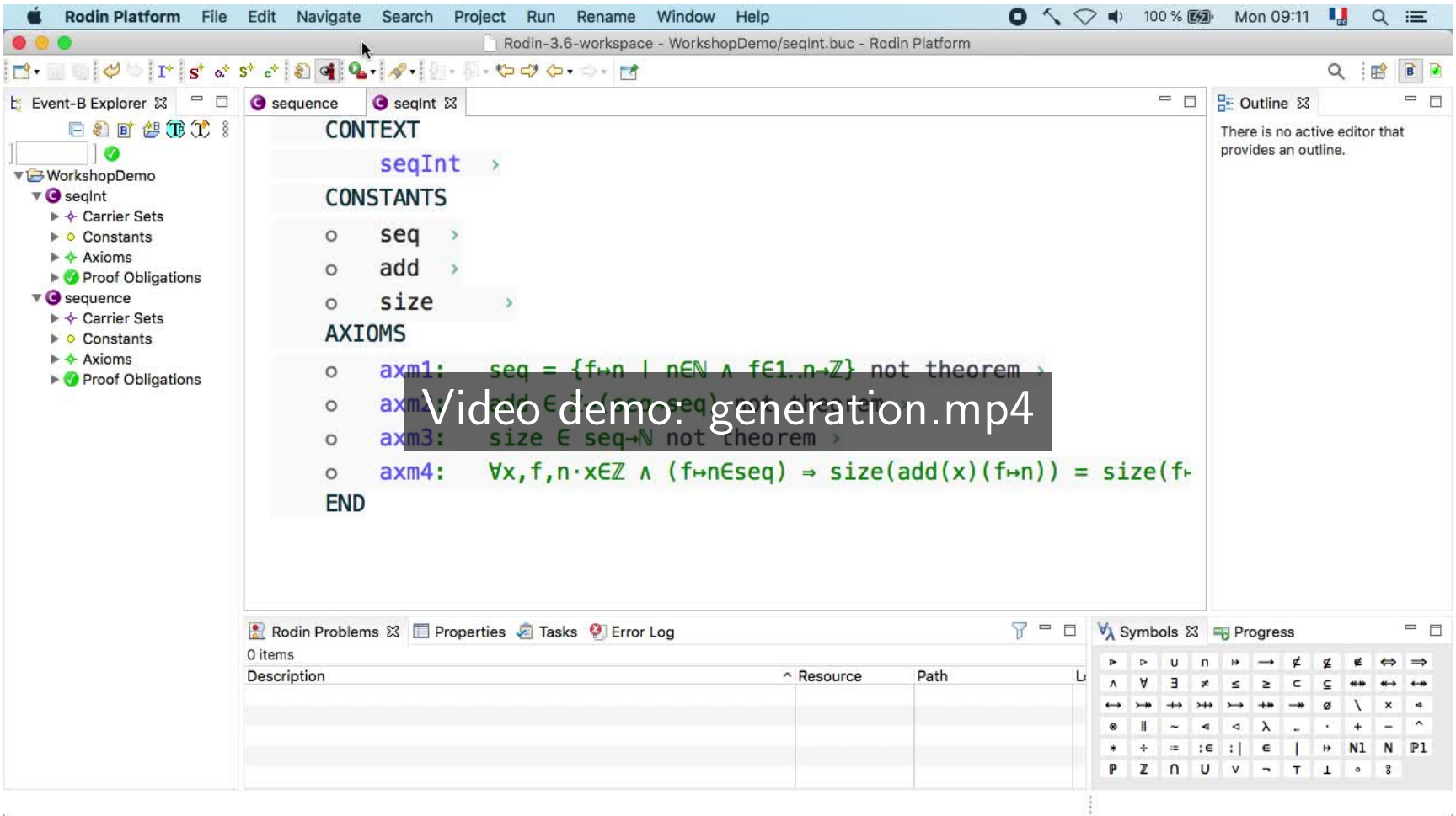
Short demo

Let's look at the previous example in Rodin:



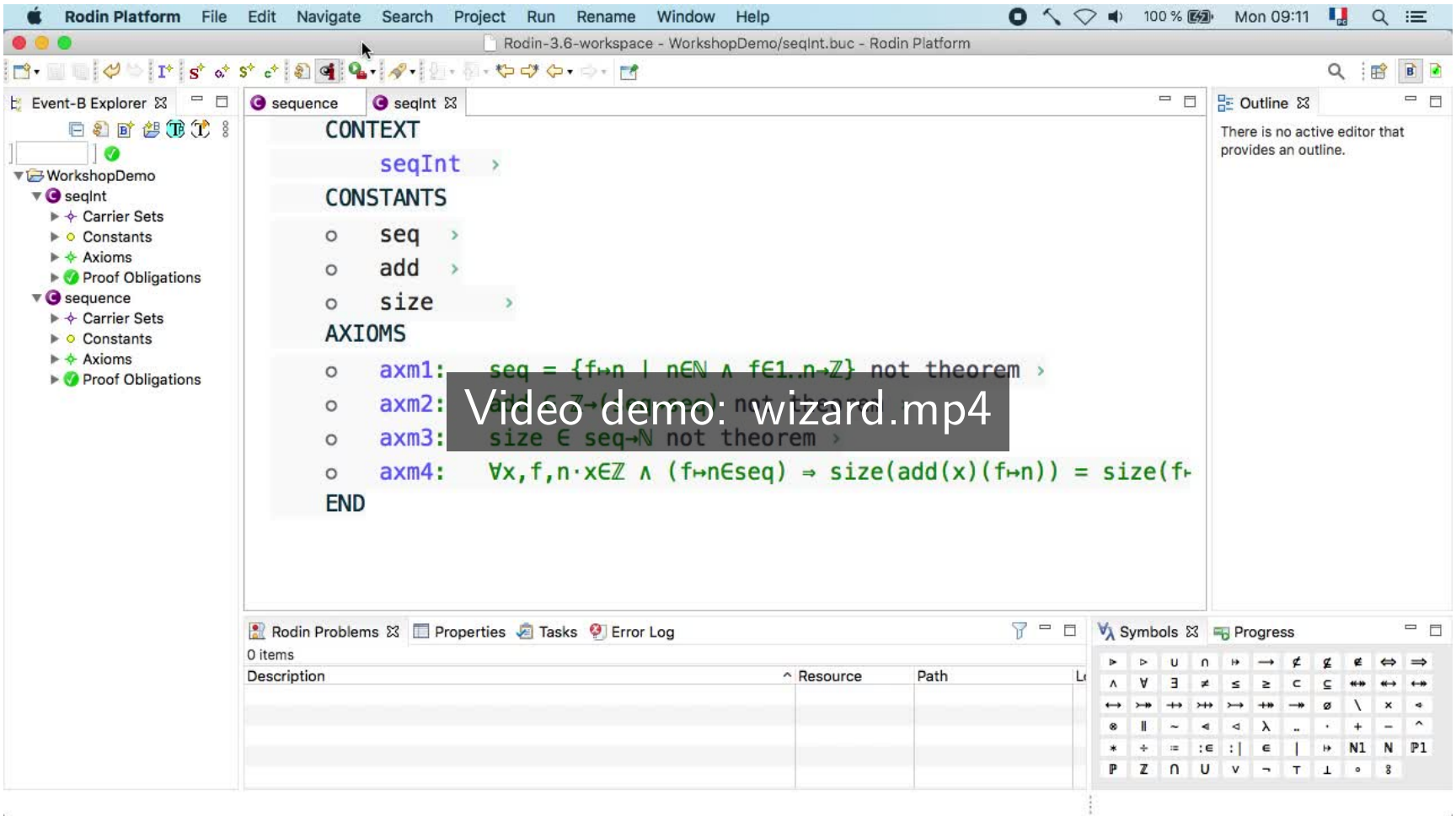
Generation

For new developments, one can provide only the instantiation in the comment box: the plug-in can generate the instantiated axiom:



Wizard

It is also possible to create instantiations through a wizard:



Types and sets

In the sequence example, S is a carrier set, used for *type* substitutions:

- S := $\mathbb{Z}$ or S := $\mathbb{Z} \times \mathbb{Z}$ are fine, for example
- S := $\mathbb{N}$ or S := $\mathbb{Z} \pfun \mathbb{Z}$ are not: they are *sets*, not types

We can adapt the generic context to use constant substitution...

context sequence
sets S_type
constants
  S
  ...
axioms
  axm1: $S \subseteq S\_type$
  ...

... and then instantiate with S_type := $\mathbb{Z}$;; S := $\mathbb{N}$, for example.

But the plug-in has type inference, so S := $\mathbb{N}$ is sufficient: authors of generic contexts have to take care of it; users do not see it.

Conclusion

The context instantiation plug-in introduces a new approach to genericity in Rodin that

- is tightly integrated in the core Rodin platform
- should be very simple to set-up and use
- can also check existing projects with duplicated code

It is currently in use by members of the EBRP project (see next talks). It should be publicly available soon.

Questions?