



| | |
|-------------|---------------|
| Edition | 0 |
| Revision | 1 |
| Page number | 14 |
| State | To be checked |

PROOF OBLIGATION GENERATOR - ***PARAMETRIZATION***

| | Redaction | Checked | Approved |
|------------------|------------------|----------------|-----------------|
| Function | Developer | | |
| Name | Lilian BURDY | | |
| Date | | | |
| Signature | | | |

| | | |
|--------|---------------|-------------------|
| Réf. : | Version : 0.1 | Date : 13/03/2017 |
|--------|---------------|-------------------|

REVISIONS

| Version | Date | Author | Comment |
|---------|------------|----------|---------|
| 0.1 | 13/03/2017 | L. Burdy | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

CONTENTS

| | |
|--|----|
| Revisions..... | 2 |
| Contents..... | 3 |
| I Introduction..... | 4 |
| I.1 Purpose..... | 4 |
| I.2 Principles..... | 4 |
| II Documents..... | 6 |
| II.1 Applicable documents..... | 6 |
| II.2 Reference documents..... | 6 |
| III IBXML..... | 7 |
| III.1 Abstraction element..... | 7 |
| III.2 Includes element..... | 7 |
| III.3 Imports element..... | 8 |
| III.4 Sees element..... | 9 |
| III.5 Extends element..... | 9 |
| III.6 Operations element..... | 9 |
| III.7 Operation_Call element..... | 9 |
| III.8 Renaming and gluing invariant..... | 9 |
| IV POXML..... | 10 |
| IV.1 Proof_Obligations..... | 10 |
| IV.2 Define..... | 10 |
| IV.3 Proof_Obligation..... | 11 |
| IV.4 Goal..... | 11 |
| IV.5 Tag..... | 11 |
| V Parametrization file format..... | 12 |
| V.1 Header..... | 12 |
| V.2 Define..... | 12 |
| V.3 Proof Obligation..... | 12 |

I INTRODUCTION

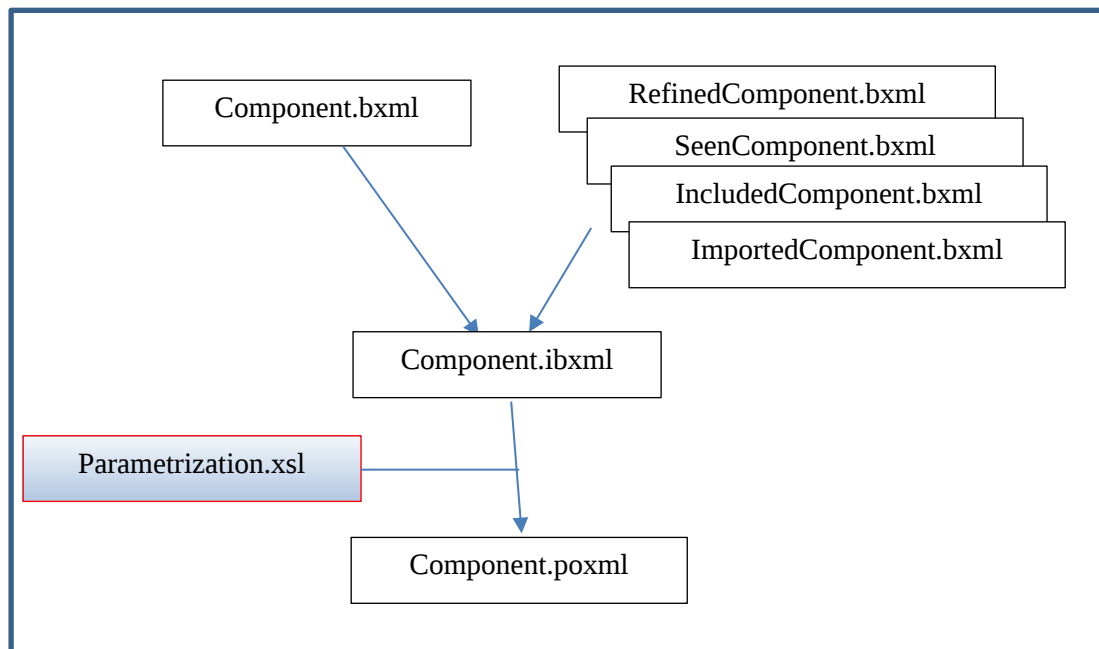
I.1 Purpose

The purpose of this document is to describe the parametrization file used to define the proof obligations to be generated by the POG.

This file is given as a parameter to the POG and defines the Proof Obligations that have to be generated. Two parametrization files are installed with the POG, one for Software B: paramGPSotware.xsl and one for Event B: paramGPSsystem.xsl.

I.2 Principles

The parametrization file is a xsl file that is applied on an ibxml file.



Figure

When a component is PO Generated, the POG applies different steps to generate the POs.

The first step consists in building an unique file, called “ibxml”, from the component bxml file and all refined, seen, included, imported component bxml files in order to have in an unique xml file all the information that are necessary for the PO Generation.

The second step consists in applying the xsl parametrization file on this ibxml file in order to generate a poxml file containing the proof obligations that will be calculated in the next steps and then translated in prover language.

This document focuses on the parametrization file but to understand how parametrization works, the first part of this document will describe the ibxml file.

II DOCUMENTS

II.1 Applicable documents

| <i>Reference</i> | <i>Version</i> | <i>Title</i> |
|---------------------------|----------------|--------------|
| BXML – Format Description | | |
| | | |
| | | |
| | | |
| | | |
| | | |

II.2 Reference documents

| <i>Reference</i> | <i>Version</i> | <i>Title</i> |
|---|----------------|-------------------------|
| http://relaxng.org/compact-20021121.html | | RELAX NG Compact Syntax |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

III IBXML

The IBXML is based on the BXML Format. In fact the BXML initial file is modified by different functions in order to obtain the IBXML file.

The format is an extension of the BXML format. This section describes only the differences between the two formats.

III.1 Abstraction element

In BXML, the abstraction element contains the name of the refined component. In IBXML, the abstraction element contains the machine elements of all the refinement hierarchy ordered by refinement.

Example:

The component:

```
REFINEMENT
  M2_r
REFINES
  M1_r
END
with
REFINEMENT
  M1_r
REFINES
  M1
END
```

is described

```
<Machine name = 'M2_r' type = 'refinement'>
  <Abstraction>
    <Machine name = 'M1_r' type = 'refinement'>
    </Machine>
    <Machine name = 'M1' type = 'abstraction'>
    </Machine>
  </Abstraction>
</Machine>
```

The refined machines are not simply inlined, some transformations on their BXML are also performed (see below).

III.2 Includes element

In BXML, the includes element contains the name of the included component with parameter valuation if exists described in a Referenced_Machine element.

In IBXML, the Referenced_Machine element is replaced by the machine element of the included component. If the included component declares parameters, those parameters are replaced by the value given in the Referenced_Machine element.

Includes element are also inlined in all other inlined components: seen, imported, included, extended.

Example:

The component:

```
REFINEMENT
  M1_r
REFINES
  M1
INCLUDES
  M2
END
with
MACHINE
  M2
INCLUDES
  M3
END
and
MACHINE
  M1
INCLUDES
  M4
END
```

is described

```
<Machine name = 'M1_r' type = 'refinement'>
  <Abstraction>
    <Machine name = 'M1' type = 'abstraction'>
      <Includes>
        <Machine name = 'M4' type = 'abstraction' />
      </Includes>
    </Machine>
  </Abstraction>
  <Includes>
    <Machine name = 'M2' type = 'abstraction' />
    <Machine name = 'M3' type = 'abstraction' />
  </Includes>
</Machine>
```

III.3 Imports element

In BXML, the imports element contains the name of the imported component with parameter valuation if exists described in a Referenced_Machine element.

IN IBXML, the Referenced_Machine element is replaced by the machine element of the imported component. If the imported component declares parameters, those parameters are replaced by the value given in the Referenced_Machine element.

Included element in imported component is also inlined.

III.4 Sees element

In BXML, the sees element contains the name of the seen component with renaming if exists described in a Referenced_Machine element.

IN IBXML, the Referenced_Machine element is replaced by the machine element of the seen component.
Included element in seen component is also inlined.

III.5 Extends element

Extends element are treated as Imports or Includes element. Moreover, the promotes element is completed.

III.6 Operations element

In IBXML, the operations element contains not only the operation element of the BXML but also implicit_operation elements corresponding to implicit operations. This element contains exactly the same information that the operation element. Those implicit operations are also added to the inlined refined component if it exists.

III.7 Operation_Call element

In IBXML, operation_call elements are completed with the operation element corresponding to the called operation. This element becomes a sub-element of operation_call element.

III.8 Renaming and gluing invariant

Variables in refinement are renamed: a suffix is added to the identifier.
Invariant of seen and imported component are duplicated to an Unrenamed_invariant where no renaming is done.
The invariant element is completed with gluing equalities between variables and renamed variables. A Renaming_Implicit_Gluing element is added as child of the machine element containing a substitution defined as the substitution that replace variables by their renamed variable. A Local_Invariant element is added as a child of the machine element containing a specific gluing invariant for local operations.

Constants are also renamed and an implicit gluing property is added to the properties element.

IV POXML

The xsl parametrization file is applied on a IBXML file and produce as output a POXML file. In order to understand how the parametrization file is built, the POXML format is described in this section.

The POXML file is a XML file, its syntax is described in Relax NG format in *poxml.rng*

```
# A RELAX NG compact syntax pattern for a poxml document.
grammar {
  include "substitution.rng"
  include "predicate.rng"
  include "expression.rng"
  include "identifier.rng"
  include "attribute.rng"

  start =
    element Proof_Obligations {
      element Define { attribute name { text }, Attribute?,
Predicate? }+,
      element Proof_Obligation {
        element Comment { text }?,
        element Tag { text }+,
        element Definition { attribute name { text } }+,
        element Hypothesis { Predicate? },
        element Goal { Predicate_or_SubCalculus }
      }+
    }+
  }
  Predicate_or_SubCalculus = Predicate
    | element Sub_Calculus { Substitution,
Predicate_or_SubCalculus}
    | element Not { Predicate_or_SubCalculus }
  Predicate |= element Tag { attribute goalTag { text }
Predicate }
}
```

This grammar includes bxml grammar files for substitutions, predicates, expressions, ...

IV.1 Proof_Obligations

The main element is a Proof_Obligations element without any attributes.

IV.2 Define

The define elements are used to factorize global hypothesis that will be referenced in proof obligations. The attribute name will be used for the referencing and must be unique for all define elements.

IV.3 Proof_Obligation

The proof_obligation elements describe the generated proof obligations. A proof obligation contains an optional comment element, some tag elements that will be used to allow user to filter the proof obligations, then proof obligation hypothesis and goals.

Hypothesis can be described as definition element referencing a define element by its name or directly by a hypothesis element containing a predicate.

IV.4 Goal

A goal element contains a specific element that can be either:

- A predicate: no specific calculation will be done in the next steps of the PO Generation.
- A Sub_Calculus element defining a substitution to apply to a predicate. This application will be calculated by the POG.
- A Not element to apply to a predicate.

IV.5 Tag

A specific Tag element with an attribute goalTag is added to the predicate definition allowing to add an annotation to a predicate. This annotation will be associated to generated goals issued from this annotated predicate and displayed to the user.

V PARAMETRIZATION FILE FORMAT

The format of the parametrization file is a xslt file transforming an IBXML file into a POXML file. In this section, as examples, extracts from the two AtelierB parametrization files are given.

V.1 Header

The header of the xslt file looks like:

```
<xsl:stylesheet version="2.0"
  xmlns:xsl=http://www.w3.org/1999/XSL/Transform
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <xsl:output method="xml"/>

  ...
</xsl:stylesheet>
```

V.2 Define

A define element looks like:

```
<!-- Define "ctx": element containing Sets and Properties from Seen
component and their included component
-->
<Define name="ctx">
  <xsl:copy-of select="Sees/Machine/Sets/*"/>
  <xsl:copy-of select="Sees/Machine/Includes/Machine/Sets/*"/>
  <xsl:copy-of select="Sees/Machine/Properties/*"/>
  <xsl:copy-of
select="Sees/Machine/Includes/Machine/Properties/*"/>
</Define>
```

V.3 Proof Obligation

The element below is the description of the proof obligation associated to the initialization of a machine (taking into account included component).

```
<!-- proof obligation for initialization
-->
<Proof_Obligation>
  <Comment>Initialisation dans une machine abstraite</Comment>
  <Tag>Initialisation</Tag>
  <Tag>Invariant</Tag>
  <!-- Hypothesis are :
    NAT = 0..MAXINT
    INT = MININT..MAXINT
    Sets and Properties from Seen component and their included
    component
    Constraints clause of the current machine
    Sets and Properties of the current machine
```

```

Sets and Properties of included components of the current machine
Invariant and Assertions of included components of the current machine
Invariant and Assertions of seen component and their included component
-->
<Definition name="B definitions"/>
<Definition name="ctx"/>
<Definition name="cst"/>
<Definition name="lprp"/>
<Definition name="inprp"/>
<Definition name="inext"/>
<Definition name="seext"/>
<!-- Goal is a substitution calculus
-->
<Goal>
  <Sub_Calculus>
    <!-- Substitution is built from Initialisations of included components
    sequenced with the initialisation of the current component
    -->
    <Nary_Sub op=";">
      <xsl:for-each select="Includes">
        <xsl:copy-of select="Machine/Initialisation/*"/>
      </xsl:for-each>
      <xsl:for-each select="Initialisation">
        <xsl:copy-of select=".*"/>
      </xsl:for-each>
    </Nary_Sub>
    <!-- Predication is an implication between conjunction of
    invariants
    and assertions of included components and the current
    component
    invariant.
    -->
    <Binary_Pred op="=>">
      <Nary_Pred>
        <xsl:attribute name="op">
          <xsl:value-of select="$and" />
        </xsl:attribute>
        <xsl:copy-of select="Includes/Machine/Invariant/*"/>
        <xsl:copy-of select="Includes/Machine/Assertions/*"/>
      </Nary_Pred>
      <Tag>
        <xsl:attribute name="goalTag">
          Invariant is preserved
        </xsl:attribute>
      </Tag>
      <Nary_Pred>
        <xsl:attribute name="op">
          <xsl:value-of select="$and" />
        </xsl:attribute>
        <xsl:copy-of select="Invariant/*"/>
      </Nary_Pred>
    </Binary_Pred>
  </Sub_Calculus>
</Goal>
</Proof_Obligation>

```

