Abstract Semantics for Java 1.4 Programs

Part I of: Verification of LePUS3/Class-Z Specifications: Sample models and Abstract Semantics for Java 1.4

Technical Report CSM-471, ISSN 1744-8050

Jonathan Nicholson, Amnon H Eden, Epameinondas Gasparis
Department of Computer Science, University of Essex, United Kingdom

31 December 2007

See also: Part II, Verification of LePUS3/Class-Z Specifications

Abstract

This document is a compendium of examples that describe the entities and relations that represent the abstract semantics (finite structures) of programs in the Java™ programming language. It is designed to provide further explanation to the definitions given in the LePUS3 and Class-Z Reference manual [Eden et al. 2007].

To remind the reader, a finite structure is a simplified ('abstracted') picture of the program, which 'flattens' the knotty structure and syntax of the source code into a set of primitive entities (also called entities of dimension 0) and relations. Essentially, each finite structure can be viewed as a relational database, or a set of tables which contain tuples of entities ('records'). Below we list a few sample Java programs and the finite structure that represents them.

See also:

- LePUS3 and Class-Z Reference manual [Eden et al. 2007]
  - Definition VII: Unary relation
  - Definition VIII: Binary relation
- Part II, Verification of LePUS3/Class-Z Specifications [Nicholson et al. 2007]

Table of contents

1. Classes

In the abstract semantics of the Java programming language, each entity in the unary relation $\underline{Class}$ (also called "a class of dimension 0") stands for one of the static types mentioned in the program which, in Java, includes classes (`java.lang.Object`), interfaces (`java.util.Collection`), primitive types (`int`), and array types (`int[]`). Following are some examples that should demonstrate the correlation between types in Java programs and the entities which represent them.

Example 1:

Program

```
public class A {
 int integer;
 int[] array = new int[5];
 class inner {
  class nestedInner { ... }
 }
}

public interface B { ... }
```

Partial finite structure

$$\{\langle \underline{A} \rangle, \langle \underline{int} \rangle, \langle \underline{int[]} \rangle, \langle \underline{A.inner} \rangle, \langle \underline{A.inner.nestedInner} \rangle, \langle \underline{B} \rangle\} = \underline{Class}$$

$$\{...\} = \underline{Method}$$

$$\{...\} = \underline{Signature}$$

...

2. Methods and signatures

Each entity in the unary relation $Method$ (also called "a method of dimension 0") stands for exactly one method in a Java Program.

Each entity in the unary relation $Signature$ (also called "a signature of dimension 0") stands for the signature (name and argument types) of one or more of the methods in a Java Program.

The pair $\langle \texttt{sig}, \texttt{mth} \rangle$ in the binary relation $SignatureOf$ indicates that the method $\texttt{mth}$ has the signature $\texttt{sig}$. Since every method in a Java program has exactly one signature, each method is associated with exactly one signature.

The relation $Member$ (see below) associates methods with classes, and the relation $Inheritable$ (see below) indicates that a method is accessible from subclasses.

The compiler adds the empty constructor if one has not been provided.

Example 2:

Program

```
public class Cls {
 public void mth() { ... }
}
```

Complete finite structure

$$\{\langle \texttt{Cls} \rangle\} = Class$$

$$\{\langle \texttt{Cls.mth()} \rangle, \langle \texttt{Cls.Cls()} \rangle\} = Method$$

$$\{\langle \texttt{mth()} \rangle, \langle \texttt{Cls()} \rangle\} = Signature$$

$$\{\langle \texttt{mth()}, \texttt{Cls.mth()} \rangle, \langle \texttt{Cls()}, \texttt{Cls.Cls()} \rangle\} = SignatureOf$$

$$\{\langle \texttt{Cls}, \texttt{Cls.mth()} \rangle, \langle \texttt{Cls}, \texttt{Cls.Cls()} \rangle\} = Member$$

$$\{\langle \texttt{Cls.mth()} \rangle\} = Inheritable$$

Example 3:

Program

```
public class A {
 public void t() { ... }
 public void z() { ... }
}

public interface B {
 public void s();
}

public class C
   extends A implements B {
 public void s() { ... }
 public void s(int i) { ... }
 public void t() { ... }
}
```

Partial finite structure

$$\{\ldots\} = \underline{Class}$$

$$\begin{aligned}\{&\langle \underline{\texttt{A.A()}}\rangle,\\ &\langle \underline{\texttt{A.t()}}\rangle,\\ &\langle \underline{\texttt{A.z()}}\rangle,\\ &\langle \underline{\texttt{B.B()}}\rangle,\\ &\langle \underline{\texttt{B.s()}}\rangle,\\ &\langle \underline{\texttt{C.C()}}\rangle,\\ &\langle \underline{\texttt{C.s()}}\rangle,\\ &\langle \underline{\texttt{C.s(int)}}\rangle,\\ &\langle \underline{\texttt{C.t()}}\rangle\}\end{aligned} = \underline{Method}$$

$$\begin{aligned}\{&\langle \underline{\texttt{A()}}\rangle,\\ &\langle \underline{\texttt{B()}}\rangle,\\ &\langle \underline{\texttt{C()}}\rangle,\\ &\langle \underline{\texttt{t()}}\rangle,\\ &\langle \underline{\texttt{z()}}\rangle,\\ &\langle \underline{\texttt{s()}}\rangle,\\ &\langle \underline{\texttt{s(int)}}\rangle\}\end{aligned} = \underline{Signature}$$

$$\begin{aligned}\{&\langle \underline{\texttt{A()}},\underline{\texttt{A.A()}}\rangle,\\ &\langle \underline{\texttt{t()}},\underline{\texttt{A.t()}}\rangle,\\ &\langle \underline{\texttt{z()}},\underline{\texttt{A.z()}}\rangle,\\ &\langle \underline{\texttt{B()}},\underline{\texttt{B.B()}}\rangle,\\ &\langle \underline{\texttt{s()}},\underline{\texttt{B.s()}}\rangle,\\ &\langle \underline{\texttt{C()}},\underline{\texttt{C.C()}}\rangle,\\ &\langle \underline{\texttt{s()}},\underline{\texttt{C.s()}}\rangle,\\ &\langle \underline{\texttt{s(int)}},\underline{\texttt{C.s(int)}}\rangle,\\ &\langle \underline{\texttt{t()}},\underline{\texttt{C.t()}}\rangle\}\end{aligned} = \underline{SignatureOf}$$

$$\begin{aligned}\{&\langle \underline{\texttt{A}},\underline{\texttt{A.A()}}\rangle,\\ &\langle \underline{\texttt{A}},\underline{\texttt{A.t()}}\rangle,\\ &\langle \underline{\texttt{A}},\underline{\texttt{A.z()}}\rangle,\\ &\langle \underline{\texttt{B}},\underline{\texttt{B.B()}}\rangle,\\ &\langle \underline{\texttt{B}},\underline{\texttt{B.s()}}\rangle,\\ &\langle \underline{\texttt{C}},\underline{\texttt{C.C()}}\rangle,\\ &\langle \underline{\texttt{C}},\underline{\texttt{C.s()}}\rangle,\\ &\langle \underline{\texttt{C}},\underline{\texttt{C.s(int)}}\rangle,\\ &\langle \underline{\texttt{C}},\underline{\texttt{C.t()}}\rangle\}\end{aligned} = \underline{Member}$$

...

3. Abstract Relation

The contents of the unary (Definition VII) $\underline{Abstract}$ relation are $\underline{\texttt{Class}}$ (Definition III) and $\underline{\texttt{Method}}$ (Definition VI) entities. If an entity exists in this relation then the related section of program is *abstract*, for example an *Interface* is always *abstract*.

Example 4:

Program

```
public abstract class A {
 public abstract void s();
 public void t() { ... }
}

public interface B {
 public void s();
}
```

Partial finite structure

$$\{\ldots\} = \underline{Class}$$

$$\{\ldots\} = \underline{Method}$$

$$\{\ldots\} = \underline{Signature}$$

$$\{\langle \underline{\texttt{A}} \rangle, \langle \underline{\texttt{A.s()}} \rangle, \langle \underline{\texttt{B}} \rangle, \langle \underline{\texttt{B.s()}} \rangle\} = \underline{Abstract}$$

...

4. Inherit Relation

The binary (Definition VIII) $\underline{Inherit}$ relation represents inheritance, in terms of Java[tm] this means the keywords `extend` and `implements`, as well as the *subtyping* relation. In Java[tm] there is no inheritance between methods, and all classes which do not explicitly extend or implement from anything are taken to extend class `java.lang.Object`.

Example 5:

Program

```
public class A extends B
   implements C { ... }

public class B { ... }

public interface C
   extends D { ... }

public interface D { ... }
```

Partial finite structure

$$\{\ldots\} = \underline{Class}$$

$$\{\ldots\} = \underline{Method}$$

$$\{\ldots\} = \underline{Signature}$$

$$\{\langle \underline{A}, \underline{B} \rangle, \langle \underline{A}, \underline{C} \rangle, \langle \underline{B}, \underline{java.lang.Object} \rangle, \langle \underline{C}, \underline{D} \rangle, \langle \underline{D}, \underline{java.lang.Object} \rangle\} = \underline{Inherit}$$

...

5. Inheritable Relation

The contents of the unary (Definition VII) $\underline{Inheritable}$ relation are $\underline{\texttt{method}}$ entities (Definition VI) which were decorated with the *public* or *protected* access modifiers. Methods declared within *final* classes are not "inheritable" as the class its self cannot be inherited from.

Constructors are never $\underline{Inheritable}$.

Example 6:

Program

```
public class A {
 public void x() { ... }
 protected void y() { ... }
 private void z() { ... }
}

public final class B {
 public void x() { ... }
 protected void y() { ... }
 private void z() { ... }
}
```

Partial finite structure

$$\{\ldots\} = \underline{Class}$$

$$\{\ldots\} = \underline{Method}$$

$$\{\ldots\} = \underline{Signature}$$

$$\{\langle \underline{\texttt{A.x()}} \rangle, \langle \underline{\texttt{A.y()}} \rangle\} = \underline{Inheritable}$$

...

6. Member and Aggregate Relations

The binary (Definition VIII) $\underline{Member}$ relation represents the ownership one class has of its methods and fields. The domain is always the "owner", and the range is either the method or type of the field in question.

The binary (Definition VIII) $\underline{Aggregate}$ relation shows that a class has a collection/array of a specified type. The domain is always the "owner", and the range is the basic type stored in the collection/array. An $\underline{Aggregate}$ relation is added for any array, or for any field which inherits from `java.util.Collection`.

Example 7:

Program

```
public class C {
  Object o1;
  Object o2;
  String[] stringArray;
  int[][] intArray;
  Set objectSet;
}
```

Partial finite structure

$$\{\ldots\} = \underline{Class}$$

$$\{\ldots\} = \underline{Method}$$

$$\{\ldots\} = \underline{Signature}$$

$$\{\langle \underline{C}, \underline{\texttt{java.lang.Object}}\rangle, \langle \underline{C}, \underline{\texttt{String[]}}\rangle, \langle \underline{C}, \underline{\texttt{int[][]}}\rangle, \langle \underline{C}, \underline{\texttt{java.util.Set}}\rangle\} = \underline{Member}$$

$$\{\langle \underline{C}, \underline{\texttt{java.lang.String}}\rangle, \langle \underline{C}, \underline{\texttt{int}}\rangle, \langle \underline{C}, \underline{\texttt{java.lang.Object}}\rangle\} = \underline{Aggregate}$$

$$\{\langle \underline{\texttt{java.util.Set}}, \underline{\texttt{java.util.Collection}}\rangle\} = \underline{Inherit}$$

...

7. Call and Forward Relations

The binary (Definition VIII) $\underline{Call}$ and $\underline{Forward}$ relations are very similar in that both indicate to a method call within the program. An invokation of a method with the same signature as the domain, using the same objects as defined by the domains arguments, is known as a forwarding method call.

Example 8:

Program

```
public class A {
 public void s() {
  s();  // Forward
  t(1); // Call
 }
 public void t(int i) {
  t(i); // Forward
  t(1); // Call
 }
}

public class B extends A {
 public void t() {
  s();  // Call
  t(1); // Call
 }
 public void t(int i) { ... }
}
```

Partial finite structure

$$\{\ldots\} = \underline{Class}$$

$$\{\ldots\} = \underline{Method}$$

$$\{\ldots\} = \underline{Signature}$$

$$\{\langle \underline{\texttt{A.s()}}, \underline{\texttt{A.t(int)}} \rangle, \langle \underline{\texttt{A.t(int)}}, \underline{\texttt{A.t(int)}} \rangle, \langle \underline{\texttt{B.t()}}, \underline{\texttt{A.s()}} \rangle, \langle \underline{\texttt{B.t()}}, \underline{\texttt{B.t(int)}} \rangle\} = \underline{Call}$$

$$\{\langle \underline{\texttt{A.s()}}, \underline{\texttt{A.s()}} \rangle, \langle \underline{\texttt{A.t(int)}}, \underline{\texttt{A.t(int)}} \rangle\} = \underline{Forward}$$

$$\ldots$$

8. Create Relation

The binary (Definition VIII) $\underline{Create}$ relation most commonly represents the keyword *new* for class instantiation, however it also represents the initialisation of a primitive type. It is worth noting that a fields initialisation is actually performed within each the constructor(s) as in the following example.

Example 9:

Program

```
public class C {
 public Object o = new Object();

 public void s() {
  int i = 0;
  new Object();
  Object s = new String();
  put(new Integer(1));
  int[][] array = new int[5][5];
 }
 public Object put(Integer i)
    { ... }
}
```

Partial finite structure

$\{\ldots\} = \underline{Class}$

$\{\ldots\} = \underline{Method}$

$\{\ldots\} = \underline{Signature}$

$\{\langle \underline{\texttt{C.C()}}, \underline{\texttt{java.lang.Object}}\rangle, \langle \underline{\texttt{C.s()}}, \underline{\texttt{int}}\rangle, \langle \underline{\texttt{C.s()}}, \underline{\texttt{java.lang.Object}}\rangle, \langle \underline{\texttt{C.s()}}, \underline{\texttt{java.lang.String}}\rangle, \langle \underline{\texttt{C.s()}}, \underline{\texttt{java.lang.Integer}}\rangle, \langle \underline{\texttt{C.s()}}, \underline{\texttt{int[][]}}\rangle\} = \underline{Create}$

...

9. Return Relation

The binary (Definition VIII) $\underline{Return}$ represents *return* statements in the program. Statements such as "*return;*" break from further execution of the method, but does not return a value. Therefore these statements are ignored.

Example 10:

Program

```
public class C {
 Object o = new Object();
 String s = new String("");
 int i = 5;

 public Object s1() {
  if (true)
   return o;
  if (true)
   return s;
  return null;
 }
 public int s2() {
  return i;
 }
}
```

Partial finite structure

$\{\dots\} = \underline{Class}$

$\{\dots\} = \underline{Method}$

$\{\dots\} = \underline{Signature}$

$\{\langle \underline{\texttt{C.s1()}}, \underline{\texttt{java.lang.Object}}\rangle, \langle \underline{\texttt{C.s1()}}, \underline{\texttt{java.lang.String}}\rangle, \langle \underline{\texttt{C.s2()}}, \underline{\texttt{int}}\rangle\} = \underline{Return}$

...

10. Produce Relation

The binary (Definition VIII) $\underline{Produce}$ relation identifies methods which "creates" and "returns" an object during execution of the method.

Example 11:

Program

```
public class C {
 public Object s1() {
  String s = "str";
  if (true)
   return new Object();
  return s;
 }

 public int s2() {
  return 5;
  }
}
```

Partial finite structure

$\{\ldots\} = \underline{Class}$

$\{\ldots\} = \underline{Method}$

$\{\ldots\} = \underline{Signature}$

$\{\langle \underline{\texttt{C.s1()}}, \underline{\texttt{java.lang.Object}}\rangle, \langle \underline{\texttt{C.s1()}}, \underline{\texttt{java.lang.String}}\rangle, \langle \underline{\texttt{C.s2()}}, \underline{\texttt{int}}\rangle\} = \underline{Produce}$

...

References

- Amnon H. Eden, Jonathan Nicholson, Epameinondas Gasparis. “The LePUS3 and Class-Z Reference manual.” Technical report CSM-474, ISSN 1744-8050 (2007), Department of Computer Science, University of Essex. [.pdf]
- Jonathan Nicholson, Amnon H Eden, Epameinondas Gasparis. "Verification of LePUS3/Class-Z Specifications: Sample models and Abstract Semantics for Java 1.4 (Part I; Part II)." Department of Computer Science, University of Essex, Tech. Rep. CSM-471, ISSN 1744-8050 (2007). [.pdf]
- James Gosling, Bill Joy, Guy Steele, Gilad Bracha. "The Java language specification", second edition, Prentice Hall (2000). [online]

Sample Models

Part II of: Verification of LePUS3/Class-Z Specifications: Sample models and Abstract Semantics for Java 1.4

Technical Report CSM-471, ISSN 1744-8050

Jonathan Nicholson, Amnon H Eden, Epameinondas Gasparis
Department of Computer Science, University of Essex, United Kingdom

See also: Part I, Verification of LePUS3/Class-Z Specifications

Abstract

This documents demonstrates how Class-Z specifications are verified using case studies. The class of design models which satisfy each sample specification is demonstrated via one or more such design models, possibly also using one or more counter examples. As there is a far greater set of design models that do not satisfy a given specification, this document is limited to a selection of design models where verification succeeds (or fails, as specified for each).

Throughout this document we assume that entity $\underline{\mathrm{cls}}$ is the interpretation of the constant cls, that is: $\mathcal{I}(\mathrm{cls}) \equiv \underline{\mathrm{cls}}$

Table of contents

1. Terms

See the LePUS3 and Class-Z reference manuals section on Terms [Eden et al. 2007] for more information.

1.1. Class

See Definition III [Eden et al. 2007].

Example 1:

Schema

Ψ

$\mathrm{aClass} : \mathbb{CLASS}$
$\mathrm{Cls} : \mathcal{P}\mathbb{CLASS}$

Satisfied by

Example 1 : A

Design models

$$\{\langle \underline{\mathrm{aClass}} \rangle, \langle \underline{\mathrm{cls}_1} \rangle, \langle \underline{\mathrm{cls}_2} \rangle, \langle \underline{\mathrm{cls}_3} \rangle\} = \underline{Class}$$

$$\underline{\mathrm{Cls}} = \{\underline{\mathrm{cls}_1}, \underline{\mathrm{cls}_2}, \underline{\mathrm{cls}_3}\}$$

1.2. Hierarchy

See Definition IV [Eden et al. 2007].

Example 2:

Schema

Ψ

$\mathbb{H}$: HIERARCHY

Design models

Satisfied by **Example 2 : A**

$$\{\langle \underline{\mathtt{root}} \rangle, \langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle\} = \underline{Class}$$

$$\mathbb{H} = \{\underline{\mathtt{root}}, \underline{\mathtt{cls}_1}, \underline{\mathtt{cls}_2}\}$$

$$\{\langle \underline{\mathtt{cls}_1}, \underline{\mathtt{root}} \rangle, \langle \underline{\mathtt{cls}_2}, \underline{\mathtt{root}} \rangle\} = \underline{Inherit}$$

Satisfied by **Example 2 : B**

$$\{\langle \underline{\mathtt{root}} \rangle, \langle \underline{\mathtt{middle}} \rangle, \langle \underline{\mathtt{bottom}} \rangle\} = \underline{Class}$$

$$\mathbb{H} = \{\underline{\mathtt{root}}, \underline{\mathtt{middle}}, \underline{\mathtt{bottom}}\}$$

$$\{\langle \underline{\mathtt{bottom}}, \underline{\mathtt{middle}} \rangle, \langle \underline{\mathtt{middle}}, \underline{\mathtt{root}} \rangle\} = \underline{Inherit}$$

Satisfied by **Example 2 : C**

$$\{\langle \underline{\mathtt{aClass}} \rangle, \langle \underline{\mathtt{root}} \rangle, \langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle\} = \underline{Class}$$

$$\mathbb{H} = \{\underline{\mathtt{root}}, \underline{\mathtt{cls}_1}, \underline{\mathtt{cls}_2}\}$$

$$\{\langle \underline{\mathtt{cls}_1}, \underline{\mathtt{aClass}} \rangle, \langle \underline{\mathtt{cls}_2}, \underline{\mathtt{aClass}} \rangle, \langle \underline{\mathtt{aClass}}, \underline{\mathtt{root}} \rangle\} = \underline{Inherit}$$

1.3. Signature

For simplicity the () characters are omitted from both methods and signatures (in both the schemas and design models) when there are no arguments present.

See Definition V [Eden et al. 2007].

Example 3:

Schema

ψ

$aSignature : \mathbb{SIGNATURE}$

$Sig : \mathbb{P}\mathbb{SIGNATURE}$

Satisfied by

Design models

Example 3 : A

$$\{\langle \underline{aSignature}\rangle, \langle \underline{sig(Object)}\rangle, \langle \underline{sig(Integer)}\rangle, \langle \underline{sig}\rangle\} = \underline{Signature}$$

$$\underline{Sig} = \{\underline{sig(Object)}, \underline{sig(Integer)}, \underline{sig}\}$$

2. Ground formulas

See the LePUS3 and Class-Z reference manuals section on the satisfaction of ground formulas [Eden et al. 2007] for more information.

2.1. Method Relation Symbol

See Definition IX (Superimpositions) [Eden et al. 2007]

Example 4:

Schema

Ψ

$\mathrm{cls} : \mathbb{CLASS}$
$\mathrm{sig} : \mathbb{SIGNATURE}$

$Method(\mathrm{sig} \otimes \mathrm{cls})$

Satisfied by **Example 4 : A**

$\{\langle \underline{\mathrm{cls}} \rangle\}$	$= \underline{Class}$
$\{\langle \underline{\mathrm{mth}} \rangle\}$	$= \underline{Method}$
$\{\langle \underline{\mathrm{sig}} \rangle\}$	$= \underline{Signature}$
$\{\langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}} \rangle\}$	$= \underline{SignatureOf}$
$\{\langle \underline{\mathrm{cls}}, \underline{\mathrm{mth}} \rangle\}$	$= \underline{Member}$

Satisfied by **Example 4 : B**

$\{\langle \underline{\mathrm{cls}} \rangle, \langle \underline{\mathrm{superClass}} \rangle\}$	$= \underline{Class}$
$\{\langle \underline{\mathrm{mth}} \rangle\}$	$= \underline{Method}$
$\{\langle \underline{\mathrm{sig}} \rangle\}$	$= \underline{signature}$
$\{\langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}} \rangle\}$	$= \underline{SignatureOf}$
$\{\langle \underline{\mathrm{superClass}}, \underline{\mathrm{mth}} \rangle\}$	$= \underline{Member}$
$\{\langle \underline{\mathrm{mth}} \rangle\}$	$= \underline{Inheritable}$
$\{\langle \underline{\mathrm{cls}}, \underline{\mathrm{superClass}} \rangle\}$	$= \underline{Inherit}$

Not satisfied by **Example 4 : C (Counter example)**

A close inspection of This case reveals that the method $\underline{\mathrm{mth}_1}$ is overridden by a non-inheritable method $\underline{\mathrm{mth}_2}$. This is a case of method hiding. In Java this is impossible as the visibility of a method cannot be reduced.

$\{\langle \underline{\mathrm{cls}} \rangle, \langle \underline{\mathrm{superClass}_1} \rangle, \langle \underline{\mathrm{superClass}_2} \rangle\}$	$= \underline{Class}$
$\{\langle \underline{\mathrm{mth}_1} \rangle, \langle \underline{\mathrm{mth}_2} \rangle\}$	$= \underline{Method}$
$\{\langle \underline{\mathrm{sig}} \rangle\}$	$= \underline{Signature}$
$\{\langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}_1} \rangle, \langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}_2} \rangle\}$	$= \underline{SignatureOf}$
$\{\langle \underline{\mathrm{superClass}_1}, \underline{\mathrm{mth}_1} \rangle, \langle \underline{\mathrm{superClass}_2}, \underline{\mathrm{mth}_2} \rangle\}$	$= \underline{Member}$
$\{\langle \underline{\mathrm{mth}_1} \rangle\}$	$= \underline{Inheritable}$
$\{\langle \underline{\mathrm{cls}}, \underline{\mathrm{superClass}_2} \rangle, \langle \underline{\mathrm{superClass}_2}, \underline{\mathrm{superClass}_1} \rangle\}$	$= \underline{Inherit}$

2.2. Abstract Relation Symbol

Example 5:

Schema

ψ

$\mathtt{cls} : \mathbb{CLASS}$
$\mathtt{sig} : \mathbb{SIGNATURE}$

$Abstract(\mathtt{cls})$
$Abstract(\mathtt{sig} \otimes \mathtt{cls})$

Satisfied by

Example 5 : A

Design models

$\{\langle \underline{\mathtt{cls}} \rangle\} = \underline{Class}$

$\{\langle \underline{\mathtt{mth}} \rangle\} = \underline{Method}$

$\{\langle \underline{\mathtt{sig}} \rangle\} = \underline{Signature}$

$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}} \rangle\} = \underline{SignatureOf}$

$\{\langle \underline{\mathtt{cls}}, \underline{\mathtt{mth}} \rangle\} = \underline{Member}$

$\{\langle \underline{\mathtt{cls}} \rangle, \langle \underline{\mathtt{mth}} \rangle\} = \underline{Abstract}$

2.3. Inherit Relation Symbol

Example 6:

Schema

Ψ

$\mathtt{subClass}, \mathtt{superClass} : \mathbb{CLASS}$

$Inherit(\mathtt{subClass}, \mathtt{superClass})$

Satisfied by **Example 6 : A**

Design models

$\{\langle \underline{\mathtt{subClass}} \rangle, \langle \underline{\mathtt{superClass}} \rangle\} = \underline{Class}$

$\{\langle \underline{\mathtt{subClass}}, \underline{\mathtt{superClass}} \rangle\} = \underline{Inherit}$

Satisfied by **Example 6 : B**

$\{\langle \underline{\mathtt{subClass}} \rangle, \langle \underline{\mathtt{superClass}} \rangle, \langle \underline{\mathtt{aClass}} \rangle\} = \underline{Class}$

$\{\langle \underline{\mathtt{subClass}}, \underline{\mathtt{aClass}} \rangle, \langle \underline{\mathtt{aClass}}, \underline{\mathtt{superClass}} \rangle\} = \underline{Inherit}$

2.4. Member Relation Symbol

Example 7:

Schema

Ψ

$\mathtt{container}, \mathtt{field} : \mathbb{CLASS}$

$Member(\mathtt{container}, \mathtt{field})$

Design models

Satisfied by **Example 7 : A**

$\{\langle \underline{\mathtt{container}} \rangle, \langle \underline{\mathtt{field}} \rangle\} = \underline{Class}$
$\{\langle \underline{\mathtt{container}}, \underline{\mathtt{field}} \rangle\} = \underline{Member}$

Satisfied by **Example 7 : B**

this is an example for subtyping in LePUS3: the formula $Member(\mathtt{container}, \mathtt{field})$ is satisfied here by class $\mathtt{subcls}$ which inherits (is a subclass of) $\mathtt{field}$

$\{\langle \underline{\mathtt{container}} \rangle, \langle \underline{\mathtt{field}} \rangle, \langle \underline{\mathtt{subClass}} \rangle\} = \underline{Class}$
$\{\langle \underline{\mathtt{subClass}}, \underline{\mathtt{field}} \rangle\} = \underline{Inherit}$
$\{\langle \underline{\mathtt{container}}, \underline{\mathtt{subClass}} \rangle\} = \underline{Member}$

Satisfied by **Example 7 : C**

As $\underline{Aggregate} \vdash \underline{Member}$, see Example 8 : A and Example 8 : B.

2.5. Aggregate Relation Symbol

Example 8:

Schema

Ψ

container, element : CLASS

$Aggregate(\texttt{container}, \texttt{element})$

Satisfied by

Design models

Example 8 : A

$\{\langle \underline{\texttt{container}} \rangle, \langle \underline{\texttt{element}} \rangle\} = \underline{Class}$

$\{\langle \underline{\texttt{container}}, \underline{\texttt{element}} \rangle\} = \underline{Aggregate}$

Satisfied by

Example 8 : B

this is an example for subtyping in LePUS3: the formula $Aggregate(\texttt{container}, \texttt{element})$ is satisfied here by class $\texttt{subcls}$ which inherits (is a subclass of) $\texttt{element}$

$\{\langle \underline{\texttt{container}} \rangle, \langle \underline{\texttt{element}} \rangle, \langle \underline{\texttt{subClass}} \rangle\} = \underline{Class}$

$\{\langle \underline{\texttt{subClass}}, \underline{\texttt{element}} \rangle\} = \underline{Inherit}$

$\{\langle \underline{\texttt{element}}, \underline{\texttt{subClass}} \rangle\} = \underline{Aggregate}$

2.6. Call Relation Symbol

Example 9:

Schema

Ψ

$\mathtt{orig}, \mathtt{dest} : \mathbb{CLASS}$

$\mathtt{sig} : \mathbb{SIGNATURE}$

$\mathit{Call}(\mathtt{sig} \otimes \mathtt{orig}, \mathtt{sig} \otimes \mathtt{dest})$

Satisfied by **Example 9 : A**

Design models

$\{\langle \underline{\mathtt{orig}} \rangle, \langle \underline{\mathtt{dest}} \rangle\}$	$= \underline{\mathit{Class}}$
$\{\langle \underline{\mathtt{caller}} \rangle, \langle \underline{\mathtt{called}} \rangle\}$	$= \underline{\mathit{Method}}$
$\{\langle \underline{\mathtt{sig}} \rangle\}$	$= \underline{\mathit{Signature}}$
$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{caller}} \rangle\}\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{called}} \rangle\}$	$= \underline{\mathit{SignatureOf}}$
$\{\langle \underline{\mathtt{orig}}, \underline{\mathtt{caller}} \rangle\}\{\langle \underline{\mathtt{dest}}, \underline{\mathtt{called}} \rangle\}$	$= \underline{\mathit{Member}}$
$\{\langle \underline{\mathtt{caller}}, \underline{\mathtt{called}} \rangle\}$	$= \underline{\mathit{Call}}$

Satisfied by **Example 9 : B**

As $\underline{\mathit{Forward}} \vdash \underline{\mathit{Call}}$, see Example 11 : A.

Example 10:

Schema

Ψ

$orig, dest : \mathbb{CLASS}$

$sig : \mathbb{SIGNATURE}$

$Call(sig \otimes orig, dest)$

Satisfied by

Example 10 : A

Design models

$\{\langle \underline{orig} \rangle, \langle \underline{dest} \rangle\}$	$= \underline{Class}$
$\{\langle \underline{caller} \rangle, \langle \underline{called} \rangle\}$	$= \underline{Method}$
$\{\langle \underline{sig} \rangle, \langle \underline{anotherSig} \rangle\}$	$= \underline{Signature}$
$\{\langle \underline{sig}, \underline{caller} \rangle\}\{\langle \underline{anotherSig}, \underline{called} \rangle\}$	$= \underline{SignatureOf}$
$\{\langle \underline{orig}, \underline{caller} \rangle\}\{\langle \underline{dest}, \underline{called} \rangle\}$	$= \underline{Member}$
$\{\langle \underline{caller}, \underline{called} \rangle\}$	$= \underline{Call}$

2.7. Forward Relation Symbol

Example 11:

Schema

Ψ

$\mathtt{orig}, \mathtt{dest} : \mathbb{CLASS}$

$\mathtt{sig} : \mathbb{SIGNATURE}$

$Forward(\mathtt{sig} \otimes \mathtt{orig}, \mathtt{sig} \otimes \mathtt{dest})$

Satisfied by

Design models

Example 11 : A

$\{\langle \underline{\mathtt{orig}} \rangle, \langle \underline{\mathtt{dest}} \rangle\}$	$= \underline{Class}$
$\{\langle \underline{\mathtt{caller}} \rangle, \langle \underline{\mathtt{called}} \rangle\}$	$= \underline{Method}$
$\{\langle \underline{\mathtt{sig}} \rangle\}$	$= \underline{Signature}$
$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{caller}} \rangle\}\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{called}} \rangle\}$	$= \underline{SignatureOf}$
$\{\langle \underline{\mathtt{orig}}, \underline{\mathtt{caller}} \rangle\}\{\langle \underline{\mathtt{dest}}, \underline{\mathtt{called}} \rangle\}$	$= \underline{Member}$
$\{\langle \underline{\mathtt{caller}}, \underline{\mathtt{called}} \rangle\}$	$= \underline{Forward}$

2.8. Create Relation Symbol

Example 12:

Schema

Ψ

$\mathtt{cls} : \mathbb{CLASS}$
$\mathtt{sig} : \mathbb{SIGNATURE}$

$Create(\mathtt{sig} \otimes \mathtt{created}, \mathtt{created})$

Design models

Satisfied by **Example 12 : A**

$\{\langle \underline{\mathtt{created}} \rangle\}$	$= \underline{Class}$
$\{\langle \underline{\mathtt{mth}} \rangle\}$	$= \underline{Method}$
$\{\langle \underline{\mathtt{sig}} \rangle\}$	$= \underline{Signature}$
$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}} \rangle\}$	$= \underline{SignatureOf}$
$\{\langle \underline{\mathtt{created}}, \underline{\mathtt{mth}} \rangle\}$	$= \underline{Member}$
$\{\langle \underline{\mathtt{mth}}, \underline{\mathtt{created}} \rangle\}$	$= \underline{Create}$

Satisfied by **Example 12 : B**

$\{\langle \underline{\mathtt{subClass}} \rangle, \langle \underline{\mathtt{created}} \rangle\}$	$= \underline{Class}$
$\{\langle \underline{\mathtt{mth}} \rangle\}$	$= \underline{Method}$
$\{\langle \underline{\mathtt{sig}} \rangle\}$	$= \underline{Signature}$
$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}} \rangle\}$	$= \underline{SignatureOf}$
$\{\langle \underline{\mathtt{created}}, \underline{\mathtt{mth}} \rangle\}$	$= \underline{Member}$
$\{\langle \underline{\mathtt{subClass}}, \underline{\mathtt{created}} \rangle\}$	$= \underline{Inherit}$
$\{\langle \underline{\mathtt{mth}}, \underline{\mathtt{subClass}} \rangle\}$	$= \underline{Create}$

Satisfied by **Example 12 : C**

As $Produce \vdash Create$, see Example 14 : A and Example 14 : B.

2.9. Return Relation Symbol

Example 13:

Schema

Ψ

returned : CLASS
sig : SIGNATURE

Return(sig⊗returned,returned)

Design models

Satisfied by **Example 13 : A**

$\{\langle \underline{\texttt{returned}} \rangle\}$	$=$ *Class*
$\{\langle \underline{\texttt{mth}} \rangle\}$	$=$ *Method*
$\{\langle \underline{\texttt{sig}} \rangle\}$	$=$ *Signature*
$\{\langle \underline{\texttt{sig}}, \underline{\texttt{mth}} \rangle\}$	$=$ *SignatureOf*
$\{\langle \underline{\texttt{returned}}, \underline{\texttt{mth}} \rangle\}$	$=$ *Member*
$\{\langle \underline{\texttt{mth}}, \underline{\texttt{returned}} \rangle\}$	$=$ *Return*

Satisfied by **Example 13 : B**

$\{\langle \underline{\texttt{returned}} \rangle, \langle \underline{\texttt{subClass}} \rangle\}$	$=$ *Class*
$\{\langle \underline{\texttt{mth}} \rangle\}$	$=$ *Method*
$\{\langle \underline{\texttt{sig}} \rangle\}$	$=$ *Signature*
$\{\langle \underline{\texttt{sig}}, \underline{\texttt{mth}} \rangle\}$	$=$ *SignatureOf*
$\{\langle \underline{\texttt{returned}}, \underline{\texttt{mth}} \rangle\}$	$=$ *Member*
$\{\langle \underline{\texttt{subClass}}, \underline{\texttt{returned}} \rangle\}$	$=$ *Inherit*
$\{\langle \underline{\texttt{mth}}, \underline{\texttt{subClass}} \rangle\}$	$=$ *Return*

Satisfied by **Example 13 : C**

As $Produce \vdash Return$, see Example 14 : A and Example 14 : B.

2.10. Produce Relation Symbol

Example 14:

Schema

Ψ

produced : CLASS
sig : SIGNATURE

Produce(sig⊗produced,produced)

Satisfied by **Example 14 : A**

Design models
$\{\langle \underline{\texttt{produced}} \rangle\} = \underline{\textit{Class}}$
$\{\langle \underline{\texttt{factory}} \rangle\} = \underline{\textit{Method}}$
$\{\langle \underline{\texttt{sig}} \rangle\} = \underline{\textit{Signature}}$
$\{\langle \underline{\texttt{sig}}, \underline{\texttt{factory}} \rangle\} = \underline{\textit{SignatureOf}}$
$\{\langle \underline{\texttt{produced}}, \underline{\texttt{factory}} \rangle\} = \underline{\textit{Member}}$
$\{\langle \underline{\texttt{factory}}, \underline{\texttt{produced}} \rangle\} = \underline{\textit{Produce}}$

Satisfied by **Example 14 : B**

Design models
$\{\langle \underline{\texttt{produced}} \rangle, \langle \underline{\texttt{subClass}} \rangle\} = \underline{\textit{Class}}$
$\{\langle \underline{\texttt{factory}} \rangle\} = \underline{\textit{Method}}$
$\{\langle \underline{\texttt{sig}} \rangle\} = \underline{\textit{Signature}}$
$\{\langle \underline{\texttt{sig}}, \underline{\texttt{factory}} \rangle\} = \underline{\textit{SignatureOf}}$
$\{\langle \underline{\texttt{produced}}, \underline{\texttt{factory}} \rangle\} = \underline{\textit{Member}}$
$\{\langle \underline{\texttt{subClass}}, \underline{\texttt{produced}} \rangle\} = \underline{\textit{Inherit}}$
$\{\langle \underline{\texttt{factory}}, \underline{\texttt{subClass}} \rangle\} = \underline{\textit{Produce}}$

3. Predicate formulas

See the LePUS3 and Class-Z reference manuals section on the satisfaction of predicate formulas (Definitions XVII, XVIII and XIX) [Eden et al. 2007] for more information.

3.1. All Predicate Symbol

See Definition XVII [Eden et al. 2007].

Example 15:

Schema

ψ

$\mathrm{cls} : \mathbb{CLASS}$
$\mathrm{sig} : \mathbb{SIGNATURE}$

$ALL(Abstract, \{ \mathrm{sig} \otimes \mathrm{cls} \})$

Satisfied by

Design models

Example 15 : A
Example 5 : A

Example 16:

Schema

ψ

$\mathrm{cls} : \mathcal{P}\mathbb{CLASS}$

$ALL(Abstract, \mathrm{cls})$

Satisfied by

Example 16 : A

Design models

$$\{\langle \underline{\mathrm{cls}_1}\rangle, \langle \underline{\mathrm{cls}_2}\rangle, \langle \underline{\mathrm{cls}_3}\rangle\} = \underline{Class}$$

$$\underline{\mathrm{cls}} = \{\underline{\mathrm{cls}_1}, \underline{\mathrm{cls}_2}, \underline{\mathrm{cls}_3}\}$$

$$\{\langle \underline{\mathrm{cls}_1}\rangle, \langle \underline{\mathrm{cls}_2}\rangle, \langle \underline{\mathrm{cls}_3}\rangle\} = \underline{Abstract}$$

Not satisfied by

Example 16 : B (Counter example)

One of the entities in $\underline{\mathrm{cls}}$ is not abstract

$$\{\langle \underline{\mathrm{cls}_1}\rangle, \langle \underline{\mathrm{cls}_2}\rangle, \langle \underline{\mathrm{cls}_3}\rangle\} = \underline{Class}$$

$$\underline{\mathrm{cls}} = \{\underline{\mathrm{cls}_1}, \underline{\mathrm{cls}_2}, \underline{\mathrm{cls}_3}\}$$

$$\{\langle \underline{\mathrm{cls}_1}\rangle, \langle \underline{\mathrm{cls}_2}\rangle\} = \underline{Abstract}$$

Example 17:

Schema

ψ

$\mathtt{cls} : \mathbb{P}\mathrm{CLASS}$
$\mathtt{sig} : \mathrm{SIGNATURE}$

$ALL(Method, \mathtt{sig} \otimes \mathtt{cls})$

Design models

Satisfied by **Example 17 : A**

$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle, \langle \underline{\mathtt{cls}_3} \rangle\}$	$= \underline{Class}$
$\underline{\mathtt{cls}}$	$= \{\underline{\mathtt{cls}_1}, \underline{\mathtt{cls}_2}, \underline{\mathtt{cls}_3}\}$
$\{\langle \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{mth}_3} \rangle\}$	$= \underline{Method}$
$\{\langle \underline{\mathtt{sig}} \rangle\}$	$= \underline{Signature}$
$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_3} \rangle\}$	$= \underline{SignatureOf}$
$\{\langle \underline{\mathtt{cls}_1}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{cls}_2}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{cls}_3}, \underline{\mathtt{mth}_3} \rangle\}$	$= \underline{Member}$

Satisfied by **Example 17 : B**

$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle, \langle \underline{\mathtt{cls}_3} \rangle\}$	$= \underline{Class}$
$\underline{\mathtt{cls}}$	$= \{\underline{\mathtt{cls}_1}, \underline{\mathtt{cls}_2}, \underline{\mathtt{cls}_3}\}$
$\{\langle \underline{\mathtt{mth}_1} \rangle\}$	$= \underline{Method}$
$\{\langle \underline{\mathtt{sig}} \rangle\}$	$= \underline{Signature}$
$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_1} \rangle\}$	$= \underline{SignatureOf}$
$\{\langle \underline{\mathtt{cls}_2}, \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_3}, \underline{\mathtt{cls}_1} \rangle\}$	$= \underline{Inherit}$
$\{\langle \underline{\mathtt{mth}_1} \rangle\}$	$= \underline{Inheritable}$
$\{\langle \underline{\mathtt{cls}_1}, \underline{\mathtt{mth}_1} \rangle\}$	$= \underline{Member}$

Not satisfied by **Example 17 : C (Counter example)**

$\underline{\mathtt{cls}_3}$, or its superclass, does not define a method with signature $\underline{\mathtt{sig}}$

$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle, \langle \underline{\mathtt{cls}_3} \rangle\}$	$= \underline{Class}$
$\underline{\mathtt{cls}}$	$= \{\underline{\mathtt{cls}_1}, \underline{\mathtt{cls}_2}, \underline{\mathtt{cls}_3}\}$
$\{\langle \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2} \rangle\}$	$= \underline{Method}$
$\{\langle \underline{\mathtt{sig}} \rangle\}$	$= \underline{Signature}$
$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_2} \rangle\}$	$= \underline{SignatureOf}$
$\{\langle \underline{\mathtt{cls}_1}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{cls}_2}, \underline{\mathtt{mth}_2} \rangle\}$	$= \underline{Member}$

3.2. Total Predicate Symbol

See Definition XVIII [Eden et al. 2007].

Example 18:

Schema

ψ

$\mathrm{cls}_1, \mathrm{cls}_2 : \mathbb{CLASS}$

$\mathrm{sig} : \mathbb{SIGNATURE}$

$\mathit{TOTAL}(\mathit{Call}, \{\mathrm{sig} \otimes \mathrm{cls}_1\}, \{\mathrm{sig} \otimes \mathrm{cls}_2$

Satisfied by **Example 18 : A**
Example 9 : A and Example 9 : B

Design models

Example 19:

Schema

ψ

$A, B : \mathcal{P}\mathrm{CLASS}$

$TOTAL(Inherit, A, B)$

Satisfied by

Example 19 : A

Design models

$\{\langle \underline{a}_1 \rangle, \langle \underline{a}_2 \rangle, \langle \underline{b}_1 \rangle, \langle \underline{b}_2 \rangle, \langle \underline{b}_3 \rangle\}$	$=$ *Class*
$\underline{A}$	$= \{\underline{a}_1, \underline{a}_2\}$
$\underline{B}$	$= \{\underline{b}_1, \underline{b}_2, \underline{b}_3\}$
$\{\langle \underline{a}_1 \rangle, \langle \underline{b}_2 \rangle\}$	$=$ *Abstract*
$\{\langle \underline{a}_1, \underline{a}_2 \rangle, \langle \underline{a}_2, \underline{b}_1 \rangle, \langle \underline{a}_2, \underline{b}_2 \rangle\}$	$=$ *Inherit*

Not satisfied by

Example 19 : B (Counter example)

There is no tuple in the required relation between one of the members of the domain and a member of the range.

$\{\langle \underline{a}_1 \rangle, \langle \underline{a}_2 \rangle, \langle \underline{b}_1 \rangle, \langle \underline{b}_2 \rangle, \langle \underline{b}_3 \rangle\}$	$=$ *Class*
$\underline{A}$	$= \{\underline{a}_1, \underline{a}_2\}$
$\underline{B}$	$= \{\underline{b}_1, \underline{b}_2, \underline{b}_3\}$
$\{\langle \underline{a}_1, \underline{b}_1 \rangle, \langle \underline{a}_1, \underline{b}_2 \rangle, \langle \underline{a}_1, \underline{b}_3 \rangle\}$	$=$ *Inherit*

Example 20:

Schema

ψ

$\mathtt{cls} : \mathbb{P}\mathrm{CLASS}$
$\mathtt{sig} : \mathrm{SIGNATURE}$

$\mathit{TOTAL}(\mathit{Call}, \mathtt{sig} \otimes \mathtt{cls}, \mathtt{sig} \otimes \mathtt{cls})$

Design models

Satisfied by **Example 20 : A**

$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle, \langle \underline{\mathtt{cls}_3} \rangle\}$	$=$	$\underline{\mathit{Class}}$
$\underline{\mathtt{cls}}$	$=$	$\{\underline{\mathtt{cls}_1}, \underline{\mathtt{cls}_2}, \underline{\mathtt{cls}_3}\}$
$\{\langle \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{mth}_3} \rangle\}$	$=$	$\underline{\mathit{Method}}$
$\{\langle \underline{\mathtt{sig}} \rangle\}$	$=$	$\underline{\mathit{Signature}}$
$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_3} \rangle\}$	$=$	$\underline{\mathit{SignatureOf}}$
$\{\langle \underline{\mathtt{cls}_1}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{cls}_2}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{cls}_3}, \underline{\mathtt{mth}_3} \rangle\}$	$=$	$\underline{\mathit{Member}}$
$\{\langle \underline{\mathtt{mth}_1}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2}, \underline{\mathtt{mth}_3} \rangle, \langle \underline{\mathtt{mth}_3}, \underline{\mathtt{mth}_1} \rangle\}$	$=$	$\underline{\mathit{Call}}$

Satisfied by **Example 20 : B**

$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle, \langle \underline{\mathtt{cls}_3} \rangle\}$	$=$	$\underline{\mathit{Class}}$
$\underline{\mathtt{cls}}$	$=$	$\{\underline{\mathtt{cls}_1}, \underline{\mathtt{cls}_2}, \underline{\mathtt{cls}_3}\}$
$\{\langle \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{mth}_3} \rangle\}$	$=$	$\underline{\mathit{Method}}$
$\{\langle \underline{\mathtt{sig}} \rangle\}$	$=$	$\underline{\mathit{Signature}}$
$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_3} \rangle\}$	$=$	$\underline{\mathit{SignatureOf}}$
$\{\langle \underline{\mathtt{cls}_1}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{cls}_2}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{cls}_3}, \underline{\mathtt{mth}_3} \rangle\}$	$=$	$\underline{\mathit{Member}}$
$\{\langle \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{mth}_3} \rangle\}$	$=$	$\underline{\mathit{Abstract}}$
$\{\langle \underline{\mathtt{mth}_1}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{mth}_1}, \underline{\mathtt{mth}_3} \rangle\}$	$=$	$\underline{\mathit{Call}}$

Satisfied by **Example 20 : C**

Every class in $\underline{\mathtt{cls}}$ defines a method with the correct signature, which are all abstract mathematically satisfying the given relation.

$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle, \langle \underline{\mathtt{cls}_3} \rangle\}$	$=$	$\underline{\mathit{Class}}$
$\underline{\mathtt{cls}}$	$=$	$\{\underline{\mathtt{cls}_1}, \underline{\mathtt{cls}_2}, \underline{\mathtt{cls}_3}\}$
$\{\langle \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{mth}_3} \rangle\}$	$=$	$\underline{\mathit{Method}}$
$\{\langle \underline{\mathtt{sig}} \rangle\}$	$=$	$\underline{\mathit{Signature}}$
$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_3} \rangle\}$	$=$	$\underline{\mathit{SignatureOf}}$
$\{\langle \underline{\mathtt{cls}_1}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{cls}_2}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{cls}_3}, \underline{\mathtt{mth}_3} \rangle\}$	$=$	$\underline{\mathit{Member}}$
$\{\langle \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{mth}_3} \rangle\}$	$=$	$\underline{\mathit{Abstract}}$

3.3. Isomorphic Predicate Symbol

See Definition XIX [Eden et al. 2007].

Example 21:

Schema

ψ

$\mathrm{cls}_1, \mathrm{cls}_2 : \mathbb{CLASS}$

$\mathit{ISOMORPHIC}(\mathit{Member}, \{\mathrm{cls}_1\}, \{\mathrm{cls}_2\})$

Satisfied by

Design models

Example 21 : A

$\{\langle \underline{\mathrm{cls}_1} \rangle, \langle \underline{\mathrm{cls}_2} \rangle\} = \underline{\mathit{Class}}$

$\{\langle \underline{\mathrm{cls}_1}, \underline{\mathrm{cls}_2} \rangle\} = \underline{\mathit{Member}}$

$\{\} = \underline{\mathit{Abstract}}$

Example 22:

Schema

ψ

$\mathtt{cls}_1, \mathtt{cls}_2 : \mathbb{CLASS}$

$\mathtt{sig} : \mathbb{SIGNATURE}$

$\mathit{ISOMORPHIC}(\mathit{Call}, \{\mathtt{sig} \otimes \mathtt{cls}_1\}, \{\mathtt{sig}($

Design models

Satisfied by **Example 22 : A**

$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle\}$	$= \underline{\mathit{Class}}$
$\{\langle \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2} \rangle\}$	$= \underline{\mathit{Method}}$
$\{\langle \underline{\mathtt{sig}} \rangle\}$	$= \underline{\mathit{Signature}}$
$\{\langle \underline{\mathtt{sig}} \rangle, \langle \underline{\mathtt{mth}_1} \rangle\} \{\langle \underline{\mathtt{sig}} \rangle, \langle \underline{\mathtt{mth}_2} \rangle\}$	$= \underline{\mathit{SignatureOf}}$
$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{mth}_1} \rangle\} \{\langle \underline{\mathtt{cls}_2} \rangle, \langle \underline{\mathtt{mth}_2} \rangle\}$	$= \underline{\mathit{Member}}$
$\{\}$	$= \underline{\mathit{Abstract}}$
$\{\langle \underline{\mathtt{mth}_1}, \underline{\mathtt{mth}_2} \rangle\}$	$= \underline{\mathit{Call}}$

Satisfied by **Example 22 : B**

$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle\}$	$= \underline{\mathit{Class}}$
$\{\langle \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2} \rangle\}$	$= \underline{\mathit{Method}}$
$\{\langle \underline{\mathtt{sig}} \rangle\}$	$= \underline{\mathit{Signature}}$
$\{\langle \underline{\mathtt{sig}} \rangle, \langle \underline{\mathtt{mth}_1} \rangle\} \{\langle \underline{\mathtt{sig}} \rangle, \langle \underline{\mathtt{mth}_2} \rangle\}$	$= \underline{\mathit{SignatureOf}}$
$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{mth}_1} \rangle\} \{\langle \underline{\mathtt{cls}_2} \rangle, \langle \underline{\mathtt{mth}_2} \rangle\}$	$= \underline{\mathit{Member}}$
$\{\langle \underline{\mathtt{mth}_2} \rangle\}$	$= \underline{\mathit{Abstract}}$
$\{\langle \underline{\mathtt{mth}_1}, \underline{\mathtt{mth}_2} \rangle\}$	$= \underline{\mathit{Call}}$

Example 23:

Schema

ψ

$\mathtt{Cls} : \mathbb{P}\mathrm{CLASS}$
$\mathtt{sig} : \mathrm{SIGNATURE}$

$\textsc{Isomorphic}(\mathit{Call}, \mathtt{sig} \otimes \mathtt{Cls}, \mathtt{sig} \otimes \mathtt{Cls})$

Design models

Satisfied by **Example 23 : A**

$$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle, \langle \underline{\mathtt{cls}_3} \rangle\} = \underline{\mathit{Class}}$$

$$\underline{\mathtt{Cls}} = \{\underline{\mathtt{cls}_1}, \underline{\mathtt{cls}_2}, \underline{\mathtt{cls}_3}\}$$

$$\{\langle \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{mth}_3} \rangle\} = \underline{\mathit{Method}}$$

$$\{\langle \underline{\mathtt{sig}} \rangle\} = \underline{\mathit{Signature}}$$

$$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_3} \rangle\} = \underline{\mathit{SignatureOf}}$$

$$\{\langle \underline{\mathtt{cls}_1}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{cls}_2}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{cls}_3}, \underline{\mathtt{mth}_3} \rangle\} = \underline{\mathit{Member}}$$

$$\{\langle \underline{\mathtt{mth}_1}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{mth}_3}, \underline{\mathtt{mth}_3} \rangle\} = \underline{\mathit{Call}}$$

Satisfied by **Example 23 : B**

$$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle, \langle \underline{\mathtt{cls}_3} \rangle\} = \underline{\mathit{Class}}$$

$$\underline{\mathtt{Cls}} = \{\underline{\mathtt{cls}_1}, \underline{\mathtt{cls}_2}, \underline{\mathtt{cls}_3}\}$$

$$\{\langle \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{mth}_3} \rangle\} = \underline{\mathit{Method}}$$

$$\{\langle \underline{\mathtt{sig}} \rangle\} = \underline{\mathit{Signature}}$$

$$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_3} \rangle\} = \underline{\mathit{SignatureOf}}$$

$$\{\langle \underline{\mathtt{cls}_1}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{cls}_2}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{cls}_3}, \underline{\mathtt{mth}_3} \rangle\} = \underline{\mathit{Member}}$$

$$\{\langle \underline{\mathtt{mth}_2} \rangle\} = \underline{\mathit{Abstract}}$$

$$\{\langle \underline{\mathtt{mth}_1}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_3}, \underline{\mathtt{mth}_3} \rangle\} = \underline{\mathit{Call}}$$

Not satisfied by **Example 23 : C (Counter example)**
There exists a method in the range that is not called by a member of the domain, and visa versa. This violates the definition of the $\textsc{Isomorphic}$ predicate.

$$\{\langle \underline{\mathtt{cls}_1} \rangle, \langle \underline{\mathtt{cls}_2} \rangle, \langle \underline{\mathtt{cls}_3} \rangle\} = \underline{\mathit{Class}}$$

$$\underline{\mathtt{Cls}} = \{\underline{\mathtt{cls}_1}, \underline{\mathtt{cls}_2}, \underline{\mathtt{cls}_3}\}$$

$$\{\langle \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{mth}_3} \rangle\} = \underline{\mathit{Method}}$$

$$\{\langle \underline{\mathtt{sig}} \rangle\} = \underline{\mathit{Signature}}$$

$$\{\langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{sig}}, \underline{\mathtt{mth}_3} \rangle\} = \underline{\mathit{SignatureOf}}$$

$$\{\langle \underline{\mathtt{cls}_1}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{cls}_2}, \underline{\mathtt{mth}_2} \rangle, \langle \underline{\mathtt{cls}_3}, \underline{\mathtt{mth}_3} \rangle\} = \underline{\mathit{Member}}$$

$$\{\langle \underline{\mathtt{mth}_1}, \underline{\mathtt{mth}_1} \rangle, \langle \underline{\mathtt{mth}_3}, \underline{\mathtt{mth}_3} \rangle\} = \underline{\mathit{Call}}$$

Example 24:

Schema

ψ

$\mathrm{Factories} : \mathbb{P}\,\mathrm{CLASS}$
$\mathrm{sig} : \mathrm{SIGNATURE}$

$\mathit{ISOMORPHIC}(\mathit{Create}, \mathrm{sig} \otimes \mathrm{Factories}, \mathrm{l}$

Design models

Satisfied by **Example 24 : A**

$\{\langle \underline{\mathrm{cls}_1}\rangle, \langle \underline{\mathrm{cls}_2}\rangle, \langle \underline{\mathrm{cls}_3}\rangle\}$	=	*Class*
$\underline{\mathrm{Factories}}$	=	$\{\underline{\mathrm{cls}_1}, \underline{\mathrm{cls}_2}, \underline{\mathrm{cls}_3}\}$
$\{\langle \underline{\mathrm{mth}_1}\rangle, \langle \underline{\mathrm{mth}_2}\rangle, \langle \underline{\mathrm{mth}_3}\rangle\}$	=	*Method*
$\{\langle \underline{\mathrm{sig}}\rangle\}$	=	*Signature*
$\{\langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}_1}\rangle, \langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}_2}\rangle, \langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}_3}\rangle\}$	=	*SignatureOf*
$\{\langle \underline{\mathrm{cls}_1}, \underline{\mathrm{mth}_1}\rangle, \langle \underline{\mathrm{cls}_2}, \underline{\mathrm{mth}_2}\rangle, \langle \underline{\mathrm{cls}_3}, \underline{\mathrm{mth}_3}\rangle\}$	=	*Member*
$\{\langle \underline{\mathrm{mth}_1}, \underline{\mathrm{cls}_1}\rangle, \langle \underline{\mathrm{mth}_2}, \underline{\mathrm{cls}_2}\rangle, \langle \underline{\mathrm{mth}_3}, \underline{\mathrm{cls}_3}\rangle\}$	=	*Create*

Satisfied by **Example 24 : B**

$\{\langle \underline{\mathrm{subClass}}\rangle, \langle \underline{\mathrm{cls}_1}\rangle, \langle \underline{\mathrm{cls}_2}\rangle, \langle \underline{\mathrm{cls}_3}\rangle\}$	=	*Class*
$\underline{\mathrm{Factories}}$	=	$\{\underline{\mathrm{cls}_1}, \underline{\mathrm{cls}_2}, \underline{\mathrm{cls}_3}\}$
$\{\langle \underline{\mathrm{mth}_1}\rangle, \langle \underline{\mathrm{mth}_2}\rangle, \langle \underline{\mathrm{mth}_3}\rangle\}$	=	*Method*
$\{\langle \underline{\mathrm{sig}}\rangle\}$	=	*Signature*
$\{\langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}_1}\rangle, \langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}_2}\rangle, \langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}_3}\rangle\}$	=	*SignatureOf*
$\{\langle \underline{\mathrm{cls}_1}, \underline{\mathrm{mth}_1}\rangle, \langle \underline{\mathrm{cls}_2}, \underline{\mathrm{mth}_2}\rangle, \langle \underline{\mathrm{cls}_3}, \underline{\mathrm{mth}_3}\rangle\}$	=	*Member*
$\{\langle \underline{\mathrm{subClass}}, \underline{\mathrm{cls}_1}\rangle\}$	=	*Inherit*
$\{\langle \underline{\mathrm{mth}_1}, \underline{\mathrm{subClass}}\rangle, \langle \underline{\mathrm{mth}_2}, \underline{\mathrm{cls}_2}\rangle, \langle \underline{\mathrm{mth}_3}, \underline{\mathrm{cls}_3}\rangle\}$	=	*Create*

Satisfied by **Example 24 : C**

$\{\langle \underline{\mathrm{subClass}}\rangle, \langle \underline{\mathrm{cls}_1}\rangle, \langle \underline{\mathrm{cls}_2}\rangle, \langle \underline{\mathrm{cls}_3}\rangle\}$	=	*Class*
$\underline{\mathrm{Factories}}$	=	$\{\underline{\mathrm{cls}_1}, \underline{\mathrm{cls}_2}, \underline{\mathrm{cls}_3}\}$
$\{\langle \underline{\mathrm{mth}_1}\rangle, \langle \underline{\mathrm{mth}_2}\rangle, \langle \underline{\mathrm{mth}_3}\rangle\}$	=	*Method*
$\{\langle \underline{\mathrm{sig}}\rangle\}$	=	*Signature*
$\{\langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}_1}\rangle, \langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}_2}\rangle, \langle \underline{\mathrm{sig}}, \underline{\mathrm{mth}_3}\rangle\}$	=	*SignatureOf*
$\{\langle \underline{\mathrm{cls}_1}, \underline{\mathrm{mth}_1}\rangle, \langle \underline{\mathrm{cls}_2}, \underline{\mathrm{mth}_2}\rangle, \langle \underline{\mathrm{cls}_3}, \underline{\mathrm{mth}_3}\rangle\}$	=	*Member*
$\{\langle \underline{\mathrm{subClass}}, \underline{\mathrm{cls}_1}\rangle\}$	=	*Inherit*
$\{\langle \underline{\mathrm{cls}_1}\rangle\}$	=	*Abstract*
$\{\langle \underline{\mathrm{mth}_1}, \underline{\mathrm{subClass}}\rangle, \langle \underline{\mathrm{mth}_2}, \underline{\mathrm{cls}_2}\rangle, \langle \underline{\mathrm{mth}_3}, \underline{\mathrm{cls}_3}\rangle\}$	=	*Create*

Example 25:

Schema

ψ

$A, B : \mathcal{P}\mathrm{CLASS}$

$\mathit{ISOMORPHIC}(\mathit{Inherit}, A, B)$

Design models

Satisfied by **Example 25 : A**

$\{\langle \underline{a}_1 \rangle, \langle \underline{a}_2 \rangle, \langle \underline{b}_1 \rangle, \langle \underline{b}_2 \rangle\} = \underline{\mathit{Class}}$
$\underline{A} = \{\underline{a}_1, \underline{a}_2\}$
$\underline{B} = \{\underline{b}_1, \underline{b}_2\}$
$\{\langle \underline{a}_1, \underline{b}_1 \rangle, \langle \underline{a}_2, \underline{b}_2 \rangle\} = \underline{\mathit{Inherit}}$

Satisfied by **Example 25 : B**

$\{\langle \underline{a}_1 \rangle, \langle \underline{a}_2 \rangle, \langle \underline{b}_1 \rangle, \langle \underline{b}_2 \rangle\} = \underline{\mathit{Class}}$
$\underline{A} = \{\underline{a}_1, \underline{a}_2\}$
$\underline{B} = \{\underline{b}_1, \underline{b}_2\}$
$\{\langle \underline{a}_1, \underline{b}_1 \rangle, \langle \underline{a}_2, \underline{b}_2 \rangle, \langle \underline{b}_1, \underline{b}_2 \rangle\} = \underline{\mathit{Inherit}}$

References

- Amnon H. Eden, Jonathan Nicholson, Epameinondas Gasparis. “The LePUS3 and Class-Z Reference manual.” Technical report CSM-474, ISSN 1744-8050 (2007), Department of Computer Science, University of Essex. [.pdf]
- Jonathan Nicholson, Amnon H Eden, Epameinondas Gasparis. "Verification of LePUS3/Class-Z Specifications: Sample models and Abstract Semantics for Java 1.4 (Part I; Part II)." Department of Computer Science, University of Essex, Tech. Rep. CSM-471, ISSN 1744-8050 (2007). [.pdf]