Beanbag: A Language for Automatic Model Inconsistency Fixing
Yingfei Xiong¹, Zhenjiang Hu², Haiyan Zhao³, Hui Song³, Masato Takeichi¹, Hong Mei³
¹Department of Mathematical Informatics, University of Tokyo, Japan
²National Institute of Informatics, Japan
³Key Laboratory of High Confidence Software Technologies, Peking University, China

1. Background
Model software system often involves models with complex relations.

A UML object diagram:

```
   o3: Entity
   o1: Entity
   o2 : Entity
   C:Container
```

A Consistency Relation:
Every persistent entity should have a container, while non-persistent entity should not.

Written in Object Constraint Language (OCL):
context Entity
inv self.persistent=true and self.container<>null or self.persistent=false and self.container=null

When we update some parts, the model becomes inconsistent. We need to update related parts to fix the inconsistency.

```
delete C ........................................... Inconsistency
```

Can we fix inconsistency automatically according to the consistency relation?

2. Fixing Behavior Ambiguity
One consistency relation may correspond to multiple fixing behaviors.

```
delete C ........................................... Inconsistency
```

```
delete o1 delete o2 ........................................... consistent
```

We need developers to specify the fixing behavior

3. Beanbag
A language that is syntactically similar to OCL, but has
• fixing semantics for fixing inconsistency
• enriched constructs for customizing fixing behavior.

A Beanbag program takes user updates and produces new updates to make model consistent.

```
def rel(entity, model):
  entity."persistent"=true and model.(entity."container")<>null or entity."persistent"=false and entity."container"=null
```

Using enriched “or” to customize fixing behavior.

```
def rel(entity, model):
  entity."persistent"=true and model.(entity."container")<>null or entity."persistent"=false and entity."container"=null
```

```
delete C delete C
```
```
1.o1.persistent <- false
2.o2.persistent <- false
```

The last two lines are swapped. “entity=null” now has higher priority.

4. Constructs in Beanbag
```
expr ::= variable
    | constant
    | expr.expr
    | not expr
    | expr=expr
    | expr and expr
    | expr or expr
    | expr->forall (v|expr)
    | expr->exists (v|expr)
    | expr->exists! (v|expr)
```

Expressions defining the same relation but have different fixing behaviors.
```
expr1=expr2 expr2=expr1
expr1 and expr2 expr2 and expr1
expr1 or expr2 expr2 or expr1
expr->exists(v|expr) expr->exists!(v|expr)
```

Visit Beanbag homepage for more information: http://www.ipl.t.u-tokyo.ac.jp/~xiong/beanbag.html