Beanbag: A Language for Automatic Model Inconsistency Fixing

Yingfei Xiong[†] Zhenjiang Hu [††] Haiyan Zhao [†††]
Hui Song [†††] Masato Takeichi [†] Hong Mei [†††]

In this poster we present Beanbag, a language for automatic model inconsistency fixing. A Beanbag program defines and checks a consistency relation over a model similarly to OCL, but the program can also be executed in a fixing mode, taking user updates on the model and producing new updates to make the model satisfy the consistency relation.

1. Motivation

Modern software development environments often involve models with complex consistency relations. For example, Figure 1 shows a UML object diagram. One consistency relation for this diagram is that every persistent entity should belong to a container while a non-persistent entity should not belong to any container.

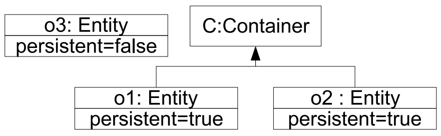

Figure 1 A UML Object

When users modify parts of the model, the consistency relation may be violated, and we need to propagate the user update to other places to fix the inconsistency. For example, when a user deletes a container, we need to either 1) change the `persistent` property of all entities in container to `false`, or 2) delete these entities directly. Manually implementing such fixing behavior is time-consuming and error-prone.

2. Approach

This poster presents the Beanbag language[2] to solve this problem. A program in Beanbag mainly defines a consistency relation, but also has a fixing semantics defining how to propagate updates to fix an inconsistency. In this way the development of inconsistency fixing is greatly eased. For example, a Beanbag program for the model in Figure 1 is shown below.

```
def relation(entity) :=
entity.persistent=true and entity.container<>null or
entity.persistent=false and entity.container=null or
entity=null
```

We can see that the program is mainly an OCL[1]-like expression describing the consistency relation. This program says an entity 1) is persistent and has a container, 2) is not persistent and has no container, or 3) does not exist.

When a container is deleted, this program can be invoked in the fixing mode for every entity. It will first try to establish the first branch of the `or` expression over the model. This attempt will fail because the container is deleted, and the program will try to establish the second and make the entity non-persistent.

If we want to instead delete the entity, we can swap the last two lines of the program. The new program will try `entity=null` first and delete the entity.

3. Evaluation

We define three basic properties for correct fixing and prove that Beanbag satisfies the three properties. In addition, we have used Beanbag to develop programs for a set of MOF and UML consistency relations. The result shows that Beanbag supports many useful fixing behavior in practice.

Reference

1) Object Management Group. Object constraint language specification 2.0. `http://www.omg.org/spec/OCL/2.0`, 2006.
2) Y. Xiong, Z. Hu, H. Zhao, H. Song, M. Takeichi, and H. Mei. Supporting automatic model inconsistency fixing. In *Proc. of 7th ESEC/FSE (to appear)*, 2009.

† Department of Mathematical Informatics, University of Tokyo. `{Yingfei_Xiong,takeichi}@mist.i.u-tokyo.ac.jp`
†† GRACE center, National Institute of Informatics. `hu@nii.ac.jp`
††† Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, China. `{zhhy,songhui06,meih}@sei.pku.edu.cn`