

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

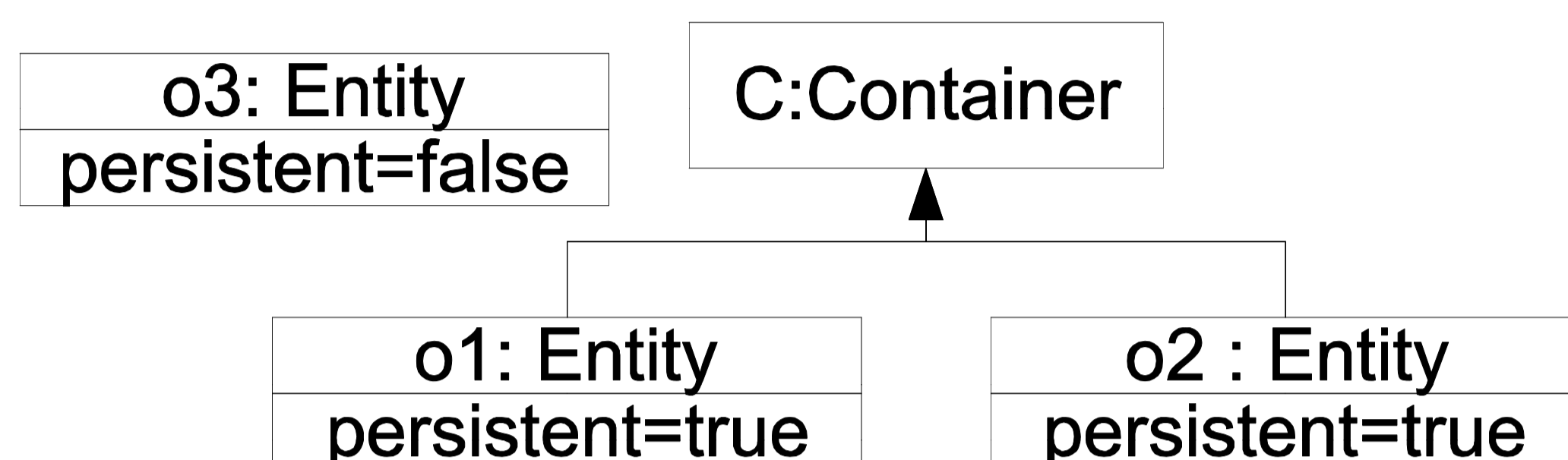
²National Institute of Informatics, Japan

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:



A Consistency Relation:

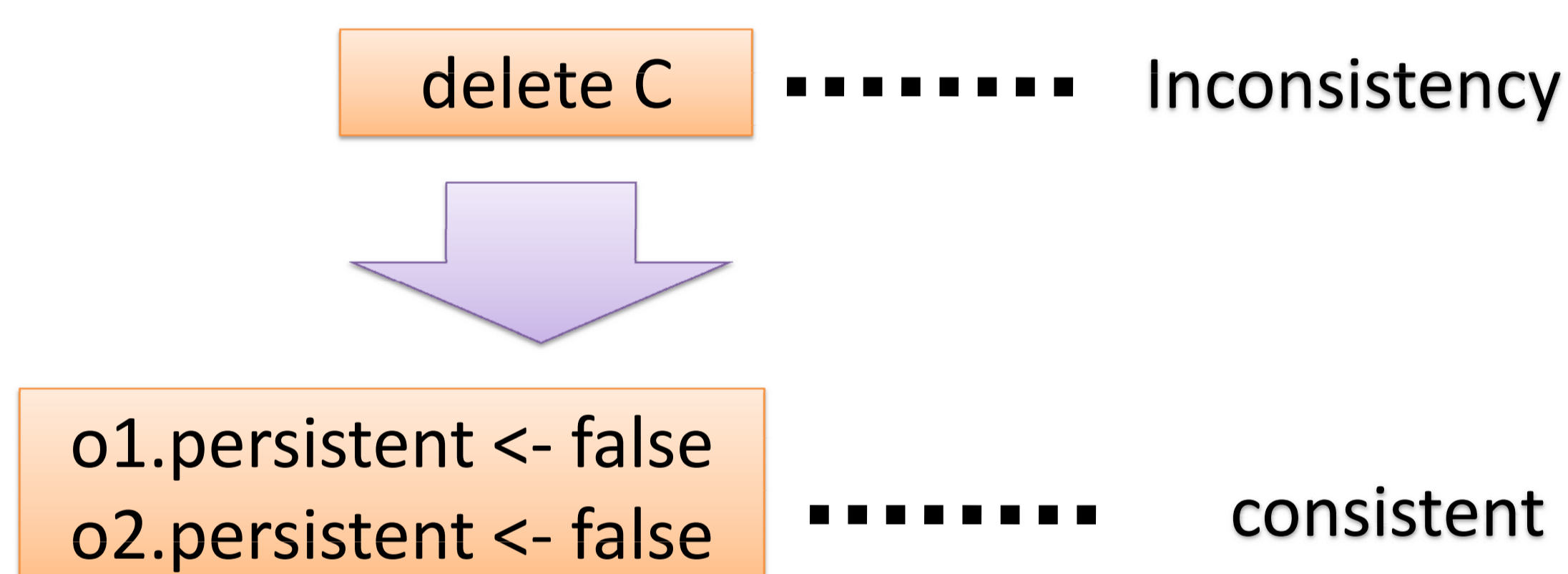
Every persistent entity should have a container, while non-persistent entity should not.

Written in Object Constraint Language (OCL):

context Entity

inv self.persistent=true and self.container<>null or
self.persistent=false and self.container=null

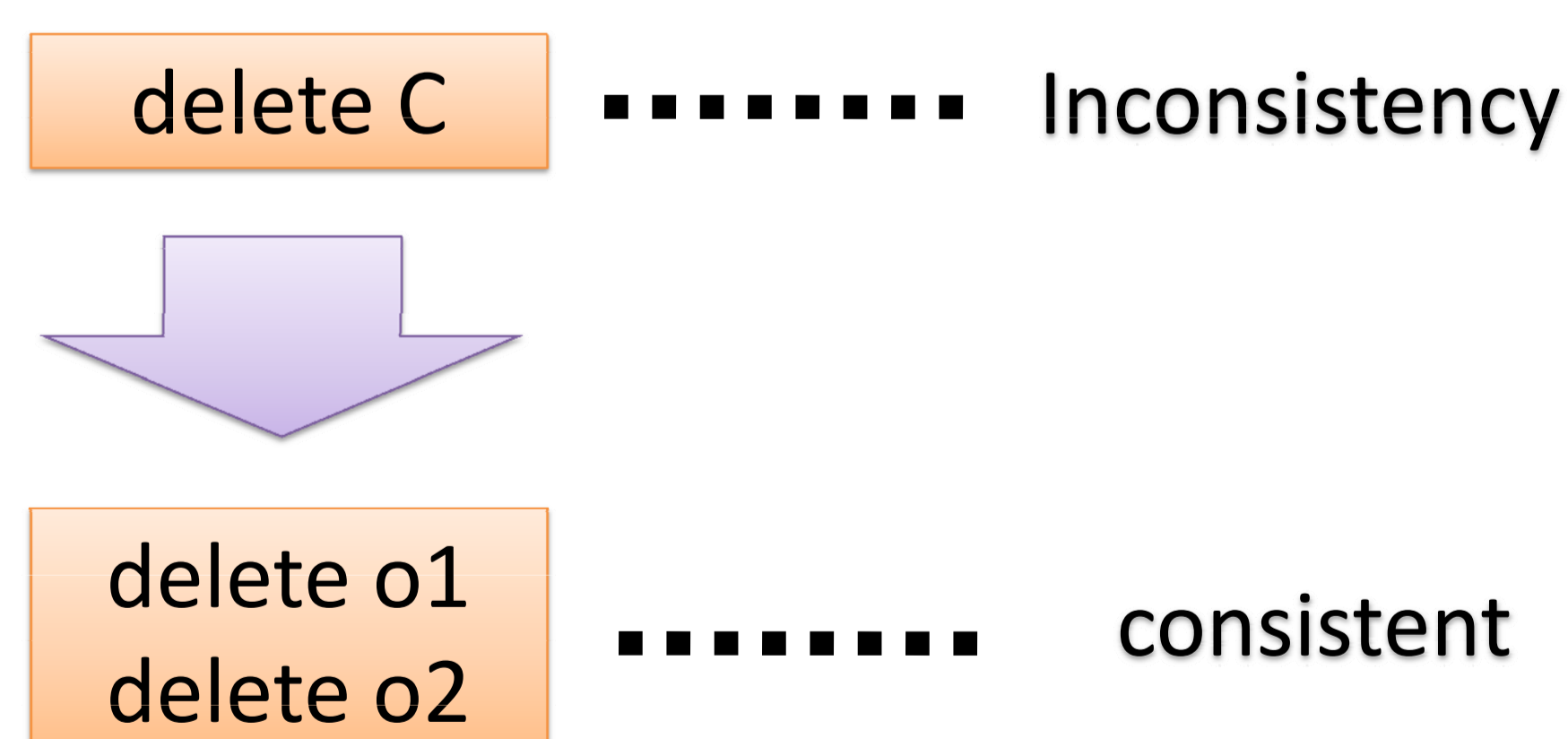
When we update some parts, the model becomes inconsistent. We need to update related parts to fix the inconsistency.



Can we fix inconsistency automatically according to the consistency relation?

2. Fixing Behavior Ambiguity

One consistency relation may correspond to multiple fixing behaviors.



We need developers to specify the fixing behavior

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*, pp.315-324, 2009.

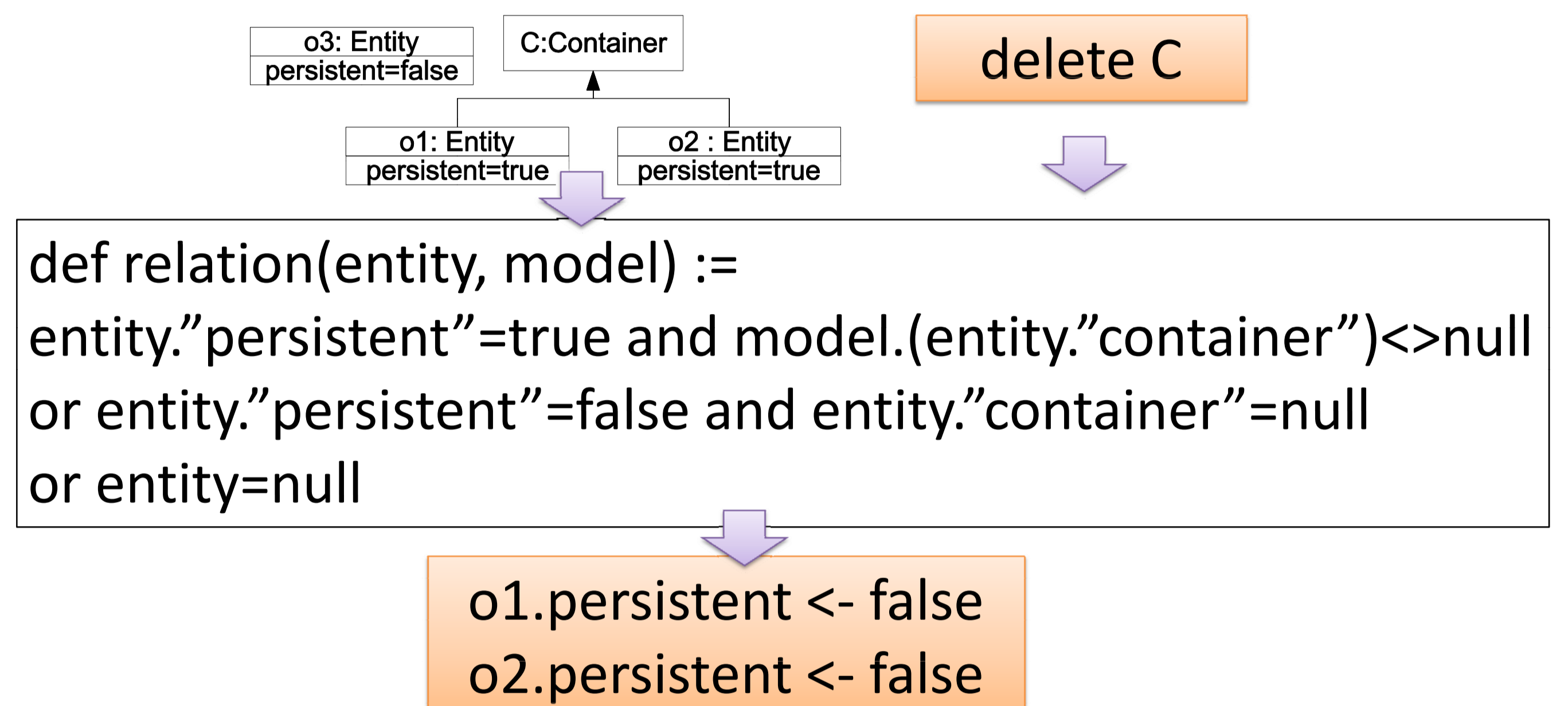
Visit Beanbag homepage for more information: <http://www.ipl.t.u-tokyo.ac.jp/~xiong/beanbag.html>

3. Beanbag

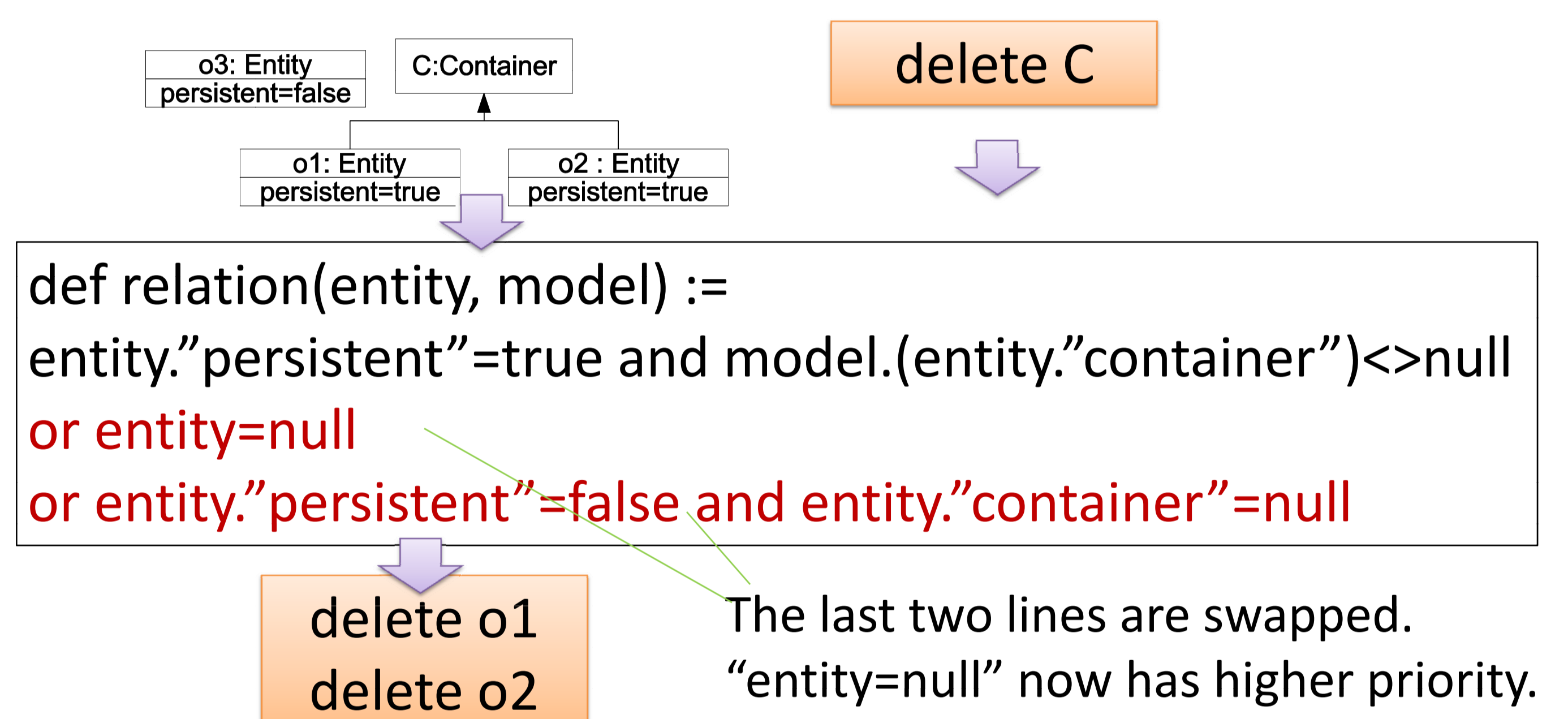
A language that is syntactically similar to OCL, but has

- *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.



Using enriched “or” to customize fixing behavior.



4. Constructs in Beanbag

Grammar of 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)