OPUS: a Calculus for Modelling Object-Oriented Concepts ERRATA

Tom Mens, Kim Mens, Patrick Steyaert

Department of Computer Science
Vrije Universiteit Brussel
Pleinlaan 2, B-1050 Brussels, BELGIUM
E-mail: { tommens@is1 **| kimmens@is1** | prsteyae@vnet3 }.vub.ac.be
WWW: http://progwww.vub.ac.be/prog/pools/opus/opus.html
Fax: +32 2 629 3495

This text contains a list of errata still present in our paper in the OOIS'94 Conference Proceedings. The changes made are printed in bold.

3.3. Reduction rules

The notation $\rightarrow$ used in the reduction rules means "...reduces in one step to...", while $\Rightarrow$ ("...reduces in **at least one step** to...") will be used in the subsequent examples to denote the transitive closure of $\rightarrow$.

3.4. Evaluation in a context

Evaluation of a method $\lambda N=E_1$ in a context leaves the method body unchanged. One might say that all **~~free~~** names in E_1 are bound by the λ.

3.5. Dealing with recursion

To simplify the examples in the rest of this paper we will abbreviate expressions of the form $\sigma_{\mathbf{self}}$ unfold:[par=E |] to σE.

4.3. Class-based inheritance

Subclasses of the point class can be created by incrementally modifying **it** for example with a MODIFIER implementing a move-method that only moves the x-coordinate while keeping the y-value unaltered.

In section 3.3 a problem still present in our approach and a possible solution are suggested:

> A problem still present in our approach is that the argument passing mechanism as proposed in this paper jeopardises encapsulation. The reason for this is that in rule 2b arguments have precedence over private methods. For example
>
> [λgetx=x λgety=y | x=1 y=2] getx:[x=2 |]
>
> yields 2 instead of the expected result 1. This problem can be solved by adding the restriction that argument names and private method names should be disjoint.

However, there is a much more simple solution to the above problem. Instead of giving arguments precedence over private methods in rule 2b, we have to do the opposite and give private methods precedence over arguments. So rule 2b becomes:

<u>Rule 2</u>: Message sending to an incrementally modified object

b) Method execution

$$(E_1 + [R \ \lambda N=E_2 \mid F]) \ N{:}E \ \rightarrow \quad \{ (\mathbf{E} + \mathbf{F}) \}(E_2) \qquad \text{if F is no Record}$$

$$\{ (\mathbf{E} + \mathbf{[\ F \mid]}) \}(E_2) \qquad \text{if F is a Record}$$

Notice that this will introduce some changes in the examples of the paper as well, but all these changes are very straightforward.