The Sesame Specification

Jonathan Busch & Günthner

Summer 25

1 Introduction

This document specifies the Bert language, associated tools and the runtime implementation of it.

2 The Language

The specification of syntax and of semantics go here.

3 The Implementation

3.1 Gabage Collection

On the language level the garbage collection will resemble the popular Mark-and-Sweepschemes used in f.e. Java. However in the implementation and in the code emitted by the compiler there is a lot of optimization possibe.

3.1.1 Avoiding Mark-and-Sweep

The difficulty in garbage collection lies in finding cycles in the reference graph. Now parts of this process can be done at compile time on the type graph. For this, some formalism will prove useful: Denote the reference digraph (w. self-reference) by (O, r) and the type digraph by (T, r'). Now the canonical projection map π will be a graph homomorphism:

$$\pi : O \to T$$

This sends and object of type t to the type t and an arrow (alt. a reference) (a, b) with objects of types t_a, t_b to (t_a, t_b).

Now we will introduce standared graph operations:
For $o \in O$, $P(o)$ ($N(o)$) denotes the previous (next) elements for o. Formally:

$$P(o) \coloneqq \{\, x \in O \mid (x, o) \in r \,\}$$
$$N(o) \coloneqq \{\, x \in O \mid (o, x) \in r \,\}$$

For reachability we must define the root set, the set of all elements which are by definition reachable, for example if they are stored in a global variable (not applicable for Bert) or if they represent a stack context.

Remark 1. *Why do we have objects representing stack contexts? They ease the fomalisation of garbage collection.*

This set we will denote by root

3.1.2 Something something Mark-and-Sweep

This will either specify the implementation of Mark-and-Sweepor an aternative scheme.

4 Tools

Here we specify usage of the `ernie` command and others.