Assignment 6
The K Machine

15-814: Types and Programming Languages
Frank Pfenning

Due Tuesday, October 29, 2019

1 Bisimulation

Task 1 (L13.1, 35 points) One unnecessary expense in the K Machine is that values v may be evaluated many times. With recursive types values v can be arbitrarily large, so we would like avoid re-evaluation. For this purpose we introduce a separate syntactic class of values w and a new expression constructor $\downarrow w$ that includes a value w as an expression. It is typed with

$$\frac{w \; \mathit{val} \quad \Gamma \vdash w : \tau}{\Gamma \vdash \downarrow w : \tau} \; \mathsf{tp/down}$$

and included in expressions with

$$\begin{array}{lrcl} \text{Expressions} & e & ::= & \ldots \mid \downarrow w \\ \text{Closed values} & w & ::= & \lambda x.\, e \mid \langle w_1, w_2 \rangle \mid \langle\,\rangle \mid i \cdot w \mid \mathsf{fold}\; w \end{array}$$

1. Update the K Machine so that the two machine states are $k \triangleright e$ and $k \triangleleft \downarrow w$. In order to avoid re-evaluation, only expressions $\downarrow w$ should be substituted for variables. Your rules should **not** appeal to the e *val* judgment but simply construct closed values w as a natural part of the machine's operation. Only show the rules for functions and pairs.

2. Establish a weak bisimulation between the machine with marked values and those without, limiting yourself to eager pairs. This means you should

   (a) Define relation R between the states in the two machines.

   (b) Prove that R is a weak bisimulation, which requires two separate properties to be shown.

   (c) Sketch the proofs of any lemmas you might need regarding the operation of each of the two machines.

3. Analyze your proof in Part 2(b) to see if you can make a statement about how the number of steps in the two machines are related.

2 Exceptions

Task 2 (L13.2, 25 points) Extend the K Machine to handle exceptions in the style of Section L11.6. There are two common technique to add exceptions.

1. We add a new form of state, $k \blacktriangleleft E$ expressing that an exception E has been raised and must be propagated or handled by k.
2. We have a pair of continuations: one is for handling normal return values, the other for handling exceptions directly. The goal is to avoid explicit unwinding of the stack because the most recent handler is directly accessible.

Write two extended versions of the K Machine following these two approaches, limiting yourself to functions, raising exceptions with raise E, and the try e_1 e_2 construct. You should make sure that your machines remain faithful to the original semantics in L11.6, but you do not need to prove it.