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 to 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

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

and included in expressions with

Expressions $\quad e ::= \ldots | \downarrow w$

Closed values $\quad w ::= \lambda x. e | \langle w_1, w_2 \rangle | \langle \rangle | i \cdot w | \text{fold } w$

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 $e \text{ val}$ judgment but simply construct closed values $w$ as a natural part of the machine’s operation. Only show the rules for functions and pairs.

The transition rules of the new machine are:

$k \triangleright \downarrow w \mapsto k \triangleleft \downarrow w$

$k \triangleright \lambda x. x \mapsto k \triangleleft (\lambda x. x)$

$k \circ (\downarrow (\lambda x. e') \_) \triangleleft \downarrow w \mapsto k \triangleright [\downarrow w / x] e'$

$k \triangleright e_1 e_2 \mapsto k \circ (_e_2) \triangleright e_1$

$k \circ (_e_2) \triangleleft \downarrow w \mapsto k \circ (\downarrow w \_) \triangleright e_2$

$k \triangleright \langle e_1, e_2 \rangle \mapsto k \circ (_e_2) \triangleright e_1$

$k \circ (_e_2) \triangleleft \downarrow w \mapsto k \circ (\downarrow w \_) \triangleright e_2$

$k \circ (\downarrow w, \_) \triangleleft \downarrow w' \mapsto k \triangleleft \langle w, w' \rangle$

$k \triangleright \text{case } e_0 (\langle x_1, x_2 \rangle \Rightarrow e) \mapsto k \circ \text{case } (\langle x_1, x_2 \rangle \Rightarrow e) \triangleright e_0$

$k \circ \text{case } (\langle x_1, x_2 \rangle \Rightarrow e) \triangleleft \langle w, w' \rangle \mapsto k \triangleright [\downarrow w / x_1, \downarrow w' / x_2] e$
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.

We can define a basic relation by simply erasing all $\downarrow$ symbols from the expressions and continuations in the new machine. We can define such erasure function by structural induction on the expressions and continuations such that erase$(e)$ and erase$(k)$ are essentially $e$ and $k$, but without any $\downarrow$, and write $k \triangleright e \ R \ k' \triangleright e'$ instead of erase$(e)$ and erase$(k)$ for the sake of brevity.

- $k \triangleright e \ R \ k' \triangleright e'$ iff either
  - $e \neq \downarrow w$ for any $w$ and $k' = \overline{k}$ and $e' = \overline{e}$, or
  - $e = \downarrow w$ for some $w$ and $k' \triangleright e' \leftrightarrow^* \overline{k} \triangleleft w$.
- $k \triangleleft e \ R \ k' \triangleleft e'$ iff $k' = \overline{k}$ and $e' = \overline{e}$.

**Theorem 1 (Weak Bisimulation for the K Machine, Part 1)**

If (in the new machine) $k_a \triangleright e_a \mapsto k_b \triangleright e_b$ and $k_a \triangleleft e_a \ R \ k'_a \triangleright e'_a$, then there are $k'_b$ and $e'_b$ such that (in the original machine) $k'_a \triangleright e'_a \mapsto^* k'_b \triangleright e'_b$ and $k_b \triangleleft e_b \ R \ k_b \triangleleft e_b$.

**Proof:** By cases on the definition of $k_a \triangleright e_a \mapsto k_b \triangleright e_b$ in the new machine.

**Case:** $k \triangleright \downarrow w \mapsto k \triangleleft \downarrow w$. Where $k_a = k_b = k$, $e_a = e_b = \downarrow w$. By definition, $k \triangleright \downarrow w \ R \ k' \triangleright e'$, where $k' \triangleright e' \leftrightarrow^* \overline{k} \triangleleft w$. It is enough to show that $k \triangleleft \downarrow w \ R \overline{k} \triangleleft w$, which is true by definition. So the proof is complete.

**Case:** $k \triangleright \langle e_1, e_2 \rangle \mapsto k \circ \langle \_, e_2 \rangle \triangleright e_1$. Where $k_a = k$, $e_a = \langle e_1, e_2 \rangle$, $k_b = k \circ \langle \_, e_2 \rangle$, and $e_b = e_1$. By definition $k \triangleright \langle e_1, e_2 \rangle \ R \overline{k} \triangleright \langle \overline{e_1}, \overline{e_2} \rangle$. In the original machine we have $\overline{k} \triangleright \langle \overline{e_1}, \overline{e_2} \rangle \mapsto \overline{k} \circ \langle \_, \overline{e_2} \rangle \triangleright \overline{e_1}$, and again by definition $k \circ \langle \_, e_2 \rangle \triangleright e_1 \ R \overline{k} \circ \langle \_, \overline{e_2} \rangle \triangleright \overline{e_1}$.

**Case:** $k \circ \langle \_, e_2 \rangle \triangleleft \downarrow w \mapsto k \circ \langle \downarrow w, \_ \rangle \triangleright e_2$. Where $k_a = k \circ \langle \_, e_2 \rangle$, $e_a = \downarrow w$, $k_b = k \circ \langle \downarrow w, \_ \rangle$ and $e_b = e_2$. By definition $k \circ \langle \_, e_2 \rangle \triangleleft \downarrow w \ R \overline{k} \circ \langle \_, \overline{e_2} \rangle \triangleleft w$. In the original machine, $\overline{k} \circ \langle \_, \overline{e_2} \rangle \triangleleft w \mapsto \overline{k} \circ \langle \downarrow w, \_ \rangle \triangleright \overline{e_2}$. And we know that $k \circ \langle \downarrow w, \_ \rangle \triangleright e_2 \ R \overline{k} \circ \langle \downarrow w, \_ \rangle \triangleright \overline{e_2}$.

**Cases:** The proof for the rest of the cases is similar.

**Theorem 2 (Weak Bisimulation for the K Machine, Part 2)**

If (in the original machine) $k'_a \triangleright e'_a \mapsto k'_b \triangleright e'_b$ and $k_a \triangleright e_a \ R \ k'_a \triangleright e'_a$, then there are $k_b$ and $e_b$, such that (in the new machine) $k_a \triangleright e_a \mapsto^* k_b \triangleright e_b$ and $k_b \triangleleft e_b \ R \ k_b \triangleleft e_b$. 
The K Machine
Sample Solution

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.

From the bisimulation theorem, we can deduce that the new machine takes less or equal steps than the original machine. Consider the proof of the first part of the theorem. In all cases except the first one both machines take same number of steps. In the first case, the new machine returns the value immediately but the original machine re-evaluate it. The number of steps it takes to reevaluate a value depends on the structure of the value.
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 techniques to add exceptions.

1. We add a new form of state, $k \downarrow E$ expressing that an exception $E$ has been raised and must be propagated or handled by $k$.

$$k \triangleright \lambda x.x \mapsto k \triangleleft (\lambda x.x)$$
$$k \circ (\lambda x.e'_1 \mathbin{-}) \triangleleft v_2 \mapsto k \triangleright [v_2/x]e'_1$$
$$k \triangleright e_1e_2 \mapsto k \circ (_e_2) \triangleright e_1$$
$$k \circ (_e_2) \triangleleft v_1 \mapsto k \circ (v_1 \mathbin{-}) \triangleright e_2$$
$$k \triangleright \text{raise } E \mapsto k \downarrow E$$

$$k \triangleright \text{try } e_1 e_2 \mapsto k \circ (\text{try } \mathbin{-} e_2) \triangleright e_1$$

$$k \circ (\text{try } \mathbin{-} e_2) \triangleleft v_1 \mapsto k \triangleleft v_1$$

$$k \circ (\text{try } \mathbin{-} e_2) \downarrow E \mapsto k \downarrow E$$

$$k \circ (_e_2) \downarrow w \mapsto k \circ (w \mathbin{-}) \triangleright e_2$$

$$k \circ (_e_2) \downarrow E \mapsto k \downarrow E$$

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.

We use two different continuations: $k_v$ for handling normal return values and $k_e$ for handling exceptions. $k_v$ is defined as the regular continuation in lecture. $k_e$ is used to store pairs of regular continuations and expressions: $k_e ::= \epsilon | k_e \circ (k_v, e)$

$$k_e; k_v \triangleright \lambda x.e \mapsto k_e; k_v \triangleleft (\lambda x.e)$$
$$k_e; k_v \circ (\lambda x.e'_1 \mathbin{-}) \triangleleft v_2 \mapsto k_e; k_v \triangleright [v_2/x]e'_1$$
$$k_e; k_v \triangleright e_1e_2 \mapsto k_e; k_v \circ (_e_2) \triangleright e_1$$
$$k_e; k_v \circ (_e_2) \triangleleft v_1 \mapsto k_e; k_v \circ (v_1 \mathbin{-}) \triangleright e_2$$
$$k_e; k_v \triangleright \text{raise } E \mapsto k_e; k_v \downarrow E$$
$$k_e; k_v \triangleright \text{try } e_1 e_2 \mapsto k_e; k_v \circ (k_v, e_2); e \triangleright e_1$$
$$k_e \circ (k'_v, e_2); k_v \triangleleft v_1 \mapsto k_e; k_v \triangleleft v_1$$
$$k_e \circ (k'_v, e_2); k_v \downarrow E \mapsto k_e; k'_v \triangleright e_2$$
$$\cdot; k_v \downarrow E \mapsto \cdot \cdot \downarrow E$$

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.