Construction & Extraction of EqMLton

Jordan Quinn

June 25, 2025

1 Construction

Given our MLton SSA, specified in [1], a construction for the CFG skeleton, S, follows. As our EqSat implementation operates on each function, we look at this structure in our SSA signature. First, we label each block with its index in the function's vector of blocks.

Then, we start at the first block and begin constructing its block in the CFG skeleton. To do this, we construct an e-graph from the statements in this block, mainly identical to the expected construction of a PEG-like e-graph. The most significant difference is that we will create a region for each node that represents the value of an assignment. When a variable already referred to by a region is used to calculate another variable, we create a new region that uses all the parameter nodes for the sub-regions, and we create a new root node that calculates the new variable using the root nodes of the sub-regions as children. This reuses the existing e-graph nodes and limits the parameters to those passed to this block.

Then, when we get to the transfer, we forget all of the regions that were not used.

After the initial construction step, it would be useful to contract sequential blocks until we are left with only merge and join points.

1. Create an empty E-Graph, G, and an empty CFG Skeleton, K
2. Create an empty map from SSA variables to region in our E-Graph, m
3. For each global in our SSA, represent them in our E-Graph and add the variable label and root E-class to m as parameterless regions
4. (Only looking at one function, f, for now) Create a $\hat{param}$ node for each function parameter to f. Add these to m as regions that take one parameter
5. Let B be a DFS traversal of all blocks in f starting from its starting block
6. For each b in B:

(a) Create a new block in the CFG skeleton with the same label and block arguments as b

(b) Create a new empty map from SSA variable label to E-Graph region

```
$G : \textsc{EGraph} \leftarrow \textsc{EGraph}()$
$B : \textsc{Set}[\textsc{SkBlock}]$
$m : \textsc{Map}[\textsc{SSAVariable}, \textsc{Region}] \leftarrow \textsc{InitMap}()$
**for** $g = \langle l : \tau \rangle$ **in** $\textsc{SSAGlobals}$ **do**
    $r \leftarrow \textsc{Region}(g, \langle\rangle)$
    $m \leftarrow m[l \mapsto r]$
    $G \leftarrow G \cup r$
**end for**
**for** $p = \langle l : \tau \rangle$ **in** $\textsc{FuncParams}$ **do**
    $r \leftarrow \textsc{Region}(\overline{param(p)}, \langle p \rangle)$
    $m \leftarrow m[l \mapsto r]$
    $G \leftarrow G \cup r$
**end for**
**for** $b$ **in** $\textsc{DFSTraversal}(\text{SSA})$ **do**
    **for** $x = \langle l : \tau \rangle$ **in** $b_{args}$ **do**
        $r \leftarrow \textsc{Region}(\overline{param(x)}, \langle x \rangle)$
        $m \leftarrow m[l \mapsto r]$
        $G \leftarrow G \cup r$
    **end for**
    **for** $s$ **in** $b_{assignments}$ **do**
        **for** $x = \langle l : \tau \rangle$ **in** $\{\langle l : \tau \rangle \in m_{keys}$ such that $l \in s\}$ **do**
            $r \leftarrow \textsc{Region}(\overline{param(x)}, \langle x \rangle)$
            $m \leftarrow m[l \mapsto r]$
            $G \leftarrow G \cup r$
        **end for**
    **end for**
    Let $t$ be $b_t$ where each variable, $x$, is replaced by $r\, r_I$ where $r = \textsc{Query}(m, x)$
    $(X, T) \leftarrow$ unzip $b_{args}$
    $B \leftarrow B \cup \textsc{SkBlock}(b_{label}, X, T, t)$
**end for**
$K_{root} \leftarrow \textsc{FuncEntryBlock}_{label}$
$K \leftarrow \textsc{Skeleton}(K_{root}, B, G, m)$
**yield** $K$
```

2 Extraction

Extraction is more complicated as we need to translate back to MLton SSA at a middle point in its optimization pipeline.

References

[1] Matthew Fluet, Henry Cejtin, Suresh Jagannathan, and Stephen Weeks. Mlton ssa representation source code. `https://github.com/MLton/mlton/blob/master/mlton/ssa/ssa-tree.fun`, 2024.