Blog on Assembly Theory

Sam Epstein

January 19, 2025

4: Bounds on Combination

Good news. Let N be the number of SECONDITIONS to be combined. Let m be the number of unique memory equivalence classes. Let s be the number of nontransient heap operations. There's a way to combine the SECONDITIONS such that the combined graph has space $O(Ns)$ and can make matches in time $O(((\log N)m + \log s)s)$. Thus, if you have the space, you can quickly determine if there is a match. This is because the constant values can be sorted. In fact, $O(Ns)$ is the worst case. On average, the space used is going to be much better.

What this means is that given the best 1000 traces,, then their SECONDITIONS can be combined together in a really efficient way. For example, if the input is a linked list then the memory equivalence classes group the lists by their size and then number values of the linked lists are actually sorted in the combined SECONDITIONS construct.

3: Floats

An interesting question is how to handle floats. The first thing to note is that a float always produces a *Const* line. However the corresponding SECONDITIONS will need work because the num values will be floats. It's unrealistic to think that two traces will have exactly the same float value. Thus it is an open question on how to modify the num values in SECONDITIONS to handle floats. One method is to specify an average error value threshold between the num values. Another method is to have special programmer code that compiles into the SECONDITIONS. Another idea is as follows. For example, say you produce the SECODE for the top 1000 float traces. When a new trace comes in, the SECHANGES of the trace with the closest float values to that of the new trace is used to compute the side-effects. Due to efficiency, you might want to test only a small fraction of the float nums.

2: Combining Conditions and Changes

I'm going to go ahead and write a followup paper entitled "Greedy Combination of Conditions and Changes in Assembly Theory". It will detail the greedy algorithm to merge the SECONDITIONS and SECHANGES constructs. This will

codify what I'm talking about in post 1. I thought the idea to be relatively simple but actually there are some cases that make the endeavor quite tricky.

1: Welcome to Assembly Theory

It apppears that there's no reason why SECONDITIONS and SECHANGES can't be combined together. This means given the best thousand traces, they'll turned into a two part code of conditions and changes and then the constructs will be compressed together by their likeness, all in $O(n \log n)$ time. Each time a trace that comes in with memory-isomorphic match to one of the thousand, then its side-effects will be computed with a single table.