CS6241 Compiler Design - Project 0

Christopher Martin (gth773s) Dan Thayer (dthayer3) Esther Goh (egoh3)

Overview

`phase0-gth773s` is a module pass. The `runOnModule(Module)` method performs analysis, removes unreachable basic blocks, then performs the same analysis again. The `print(raw_ostream, Module)` method prints the results of both analyses when `opt` is run with the `-analyze` flag.

Analysis

The `IntStats` struct accumulates data regarding average, min and max computations. `ModuleStats`, `FunctionStats`, and `BasicBlockStats` are used to aggregate statistics pertaining to a module, a function, and a basic block respectively.

1. Total number of functions and edges in static call graph

	g721	gsm	jpeg	mpeg2	rawcaudio	rawdaudio
Functions	26	94	391	115	3	3
Edges	10	89	358	221	1	1

The number of function definitions are counted using the module's function iterator, incrementing the count only when `Function->isDeclaration()` is `false` to avoid counting externally-defined functions without bodies to analyze.

To analyze the static call graph, for each function we build a `set` of the functions that can be called by it. This is done by iterating over each function's instructions. If an instruction is a `CallInst`, its called function is added to the set of functions that may be called from the current function. The size of this set for each function is summed to get the total number of call graph edges.

2. Number of basic blocks

	g721	gsm	jpeg	mpeg2	rawcaudio	rawdaudio
Min	2	2	2	2	16→14	16→14
Max	206	290	270	178	128→126	60
Average	26.8	33.0→32.1	28.78→28.76	35.17→35.15	42.7→42.0	42.7→42.0

For each function, we just iterate over its basic blocks to count them. We can see that the number of basic blocks is reduced by the optimization in every benchmark except g721 - the optimization pass left that module completely untouched.

3. Number of CFG edges

	g721	gsm	jpeg	mpeg2	rawcaudio	rawdaudio
Min	0	0	0	0	7	7
Max	137	192	192	126	41	41
Average	17.0	20.3	18.9	23.0	27.7	27.7

The control flow graph is represented as a `map<BasicBlock*, set<BasicBlock*>>`, wherein each block is mapped its set of successors. The `getCfg(Function)` method generates this map (this function will be used for other analyses as well). It iterates over a function's blocks, then iterates over each block's terminator's successors to add them to the block's successor list. The number of CFG edges in a function is the sum of the cardinalities of each set in this map.

4. Number of single-entry loops

	g721	gsm	jpeg	mpeg2	rawcaudio	rawdaudio
Min	0	0	0	0	1	1
Max	3	11→9	21	16	1	1
Average	0.38	1.1→0.68	1.23	1.0	1.0	1.0

To find single-entry loops, we generate a dominator tree using the same map-based data structure as in the previous problem (the result is a map from each basic block to the set of blocks that dominate it). First we generate the control flow graph using `getCfg`, and reverse the direction of the edges using the `transpose` method. The result is the set of predecessors for each block. The dominator data structure is initialized as follows: the entry block's only dominator is itself, and every other block is dominated by everything. We then set each block's dominator set to the intersection of the dominators of the block's predecessors, plus the block itself. This is repeated until the data structure stops changing.

Single-entry loops are detected as backedges in the dominator tree. We consider every pair of blocks `u`, `v`. If `u` dominates `v`, and (`v`, `u`) is an edge in the control flow graph, then (`v`, `u`) is a backedge.

5. Number of loop basic blocks

	g721	gsm	jpeg	mpeg2	rawcaudio	rawdaudio
Min	1	1	1	1	8→7	8→7
Max	103	145	135	89	30	30
Average	13.4	16.5→16.0	14.39→14.38	17.58→17.57	21.3→21.0	21.3→21.0

This is the number of basic blocks which belong to a cycle in the control flow graph.
The `getReach(Function)` method generates, for each block, the set of blocks that are reachable from it in the CFG. We consider every pair of blocks `u`, `v`. If `u` is reachable from `v` and `v` is reachable from `u`, then `u` and `v` both belong to a cycle.

6-7. Average number of dominators of a basic block across all functions

g721	gsm	jpeg	mpeg2	rawcaudio	rawdaudio
2.45	3.83	3.20	2.95	3.45	3.45

Using the dominator tree calculated previously, we just need to count the number of dominators The total number of dominators across the module is then divided by the total number of basic blocks to find the average. Finding the average number of blocks dominated by a basic block is the opposite of the above case, and naturally these results are the same.

Unreachable code detection and elimination

To detect unused blocks, we call `removeUnusedBlocks(Function)` on each function in the module. Again we use the result of `getReach(Function)`, which gives the reachability between every pair of blocks. Any block B that is not reachable from the entry block is removed by calling `B->eraseFromParent()`.