MO601 – Project 4
LISC
Learning Instruction Semantics from Code Generators

093311 – Alberto Arruda de Oliveira

LISC Objective

- Automated modeling of instruction semantics
- Compilers: Intermediate Language -> Assembly Code
  - GCC, LLVM, etc.
- LISC:
  - Learn from code generators
  - Automatically extract semantics
  - Test extracted semantics

Project 4 Objective

- Reproduce the completeness experiment (Table 4) of reference [1]
  - Project 3 Objective

- Evaluate the instruction lifting performance of LISC
  - Is the performance reported by [1] accurate?
  - How does the program size affects lifting performance?
  - Does the automata used impacts lifting performance?

[1] Niranjan Hasabnis and R. Sekar. 2016. Lifting Assembly to Intermediate Representation: A Novel Approach Leveraging Compilers. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS '16). ACM, New York, NY, USA, 311-324.

LISC Evaluation Completeness

- Original Methodology:
  - Testing programs P_{test} and training programs $P_{train} \subset P_{test}$
  - Build a transducer using P_{train}
  - Translate the assembly of P_{test}
- How was P_{test} chosen?
  - All x86 binaries, including kernel modules, found on Ubuntu-14.04 desktop (around 38M unique instructions)
- How was P_{train} chosen?
  - Started as *openssl-1.0.1f* + *binutils-2.22* binaries
  - Each round added a new binary

LISC Evaluation Completeness

- How was P_{test} chosen?
  - All x86 binaries, including kernel modules, found on Ubuntu-14.04 desktop (38M unique instructions)

- How was P_{train} chosen?
  - Started as *openssl-1.0.1f* + *binutils-2.22* binaries;
  - Each round added a new binary, in order: *ffmpeg-2.3.3* (With no optmizations), *glibc-2.21*, *ffmpeg-2-3-3(With optmizations), gstreamer-1.4.5, qt-5.4.1, linux-kernel-3.19, Manual instructions*.

LISC Evaluation
Completeness Metrics

- Comparison between:
  - **Exact Recall:** an instruction from P_{test} is lifted only if it also belongs to P_{train};
  - **LISC %** of instructions lifted

- From the LISC %:
  - Number of Mnemonics missing, from the total of 1187 found in x86:
    - Percentage;
    - Absolute Number;
  - Percentage of operands missing

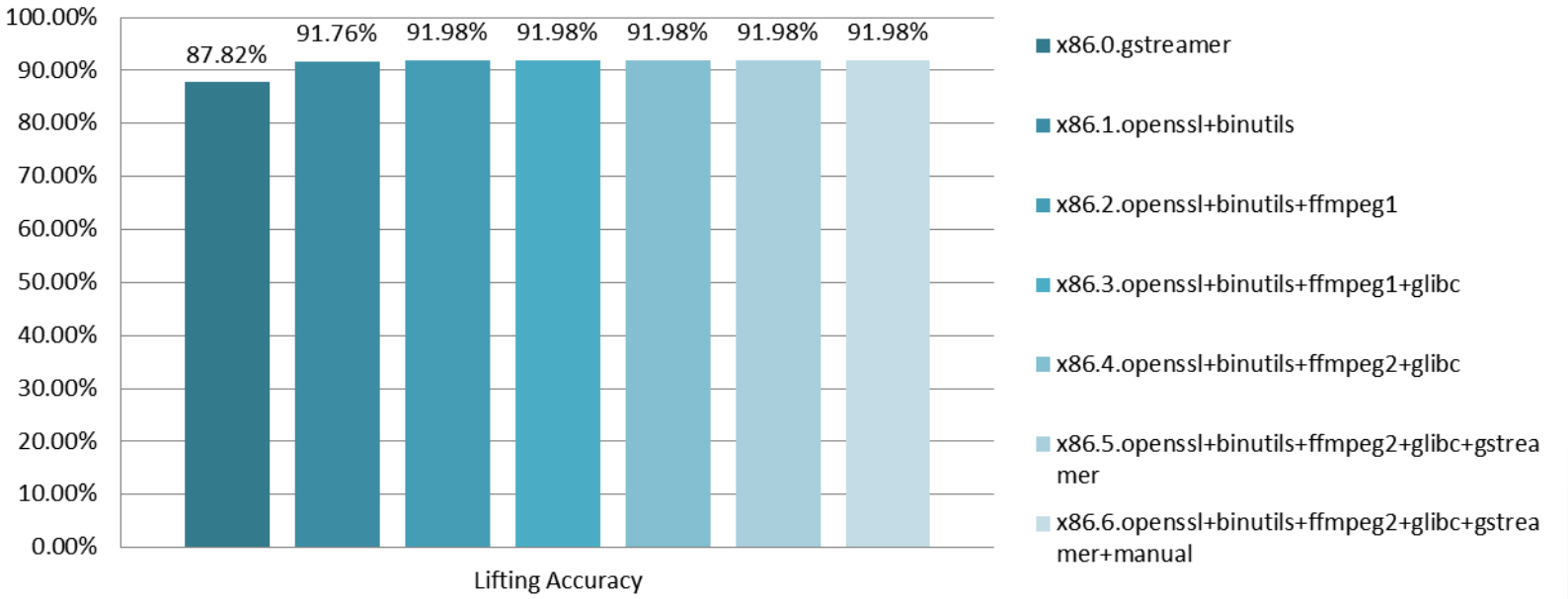
LISC Evaluation
Completeness Methodology Changes

- Made to guarantee reproducibility of results
  - P_{train} is a different set of programs (does not include QT and Linux Kernel)
  - P_{test} is a limited set of Ubuntu Server 14.04 binaries. Only binaries with less than 6MB were used, because of time constraints (Around 14M instructions total)
  - Binaries were evaluated individually
    - More transparent evaluation approach
  - Average **LISC Lifting %** computed
    - Exact Recall could not be computed

LISC Evaluation
Completeness Original Results

P_train	% Instructions Lifted		LISC (%)		Missing Mnemonics (Absolute)
	Exact Recall	LISC	Missing Mnemonics	Missing Operands	
openssl-1.0.1f + binutils-2.22	63.72	98.46	1.05	0.49	464
+ffmpeg-2.3.3 (Non Opt)	68.21	98.74	1.03	0.23	377
+glibc-2.21	68.74	98.8	1.01	0.19	346
+ffmpeg-2.3.3 (Opt)	69.07	98.89	0.88	0.23	303
+gstreamer-1.4.5	71.07	99.1	0.79	0.11	221
+qt-5.4.1	72.45	99.21	0.69	0.09	161
+linuxkern-3.19	73.97	99.49	0.44	0.07	49
+Manual	74.04	100	0	0	0

LISC Evaluation
Completeness New Results



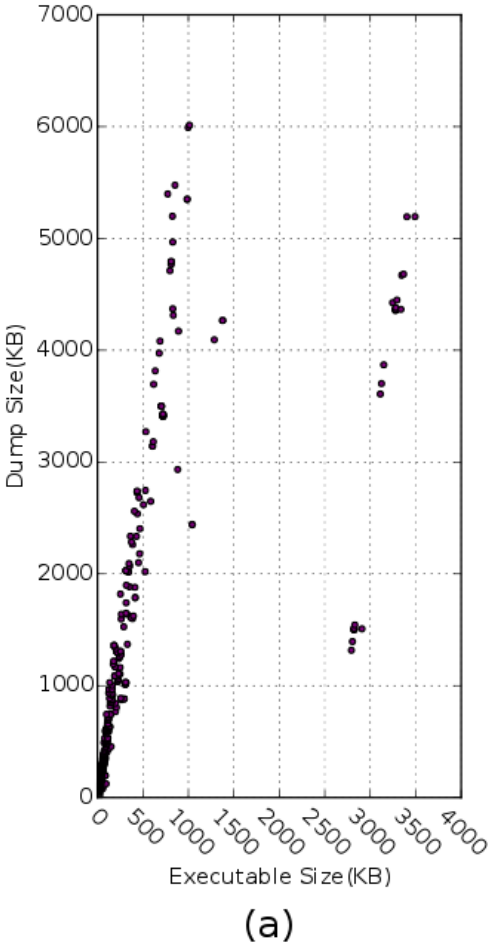
LISC Evaluation
Completeness Discussion

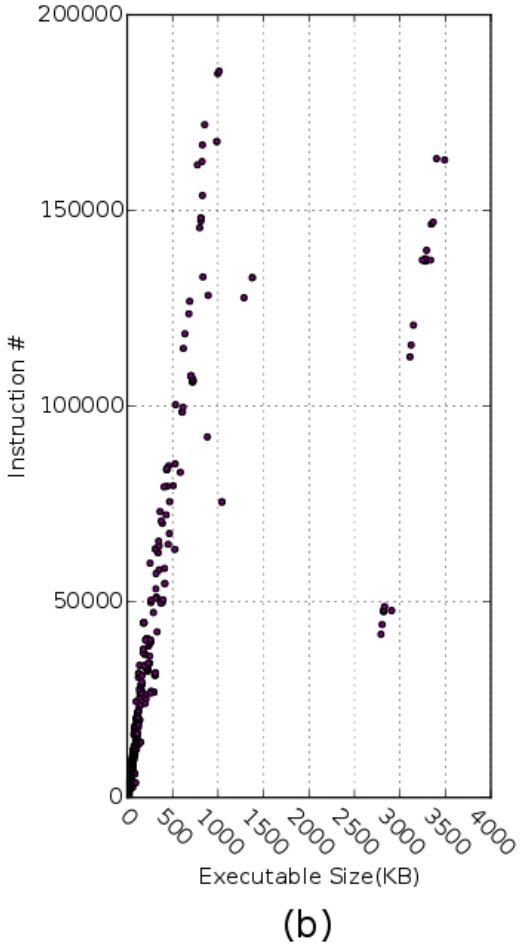
- Huge difference between Original and New results, even in comparable experiments
  - Over 10 percentual points in some cases
- New methodology is more ‘fair’, by not excluding duplicates
- Most complementary automatas do not change the final results at all
- *Ldconfig* results were very low.

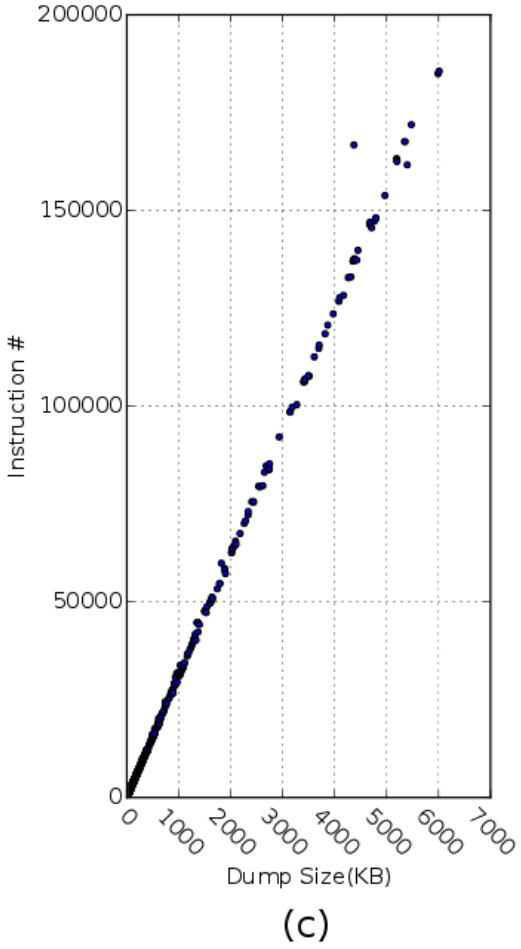
LISC Evaluation
Performance Evaluation

- Performance reporting in the original paper was lacking
  - Only graphically reports automata creation time
- Biggest burden in performance is binary lifting
  - Liner algorithm with program size
  - The authors argue 8h total to lift around 38M instructions, a fifth of which is to dump programs
  - Very different results observed in practice
- Significantly smaller number of instructions took around 13h only to lift

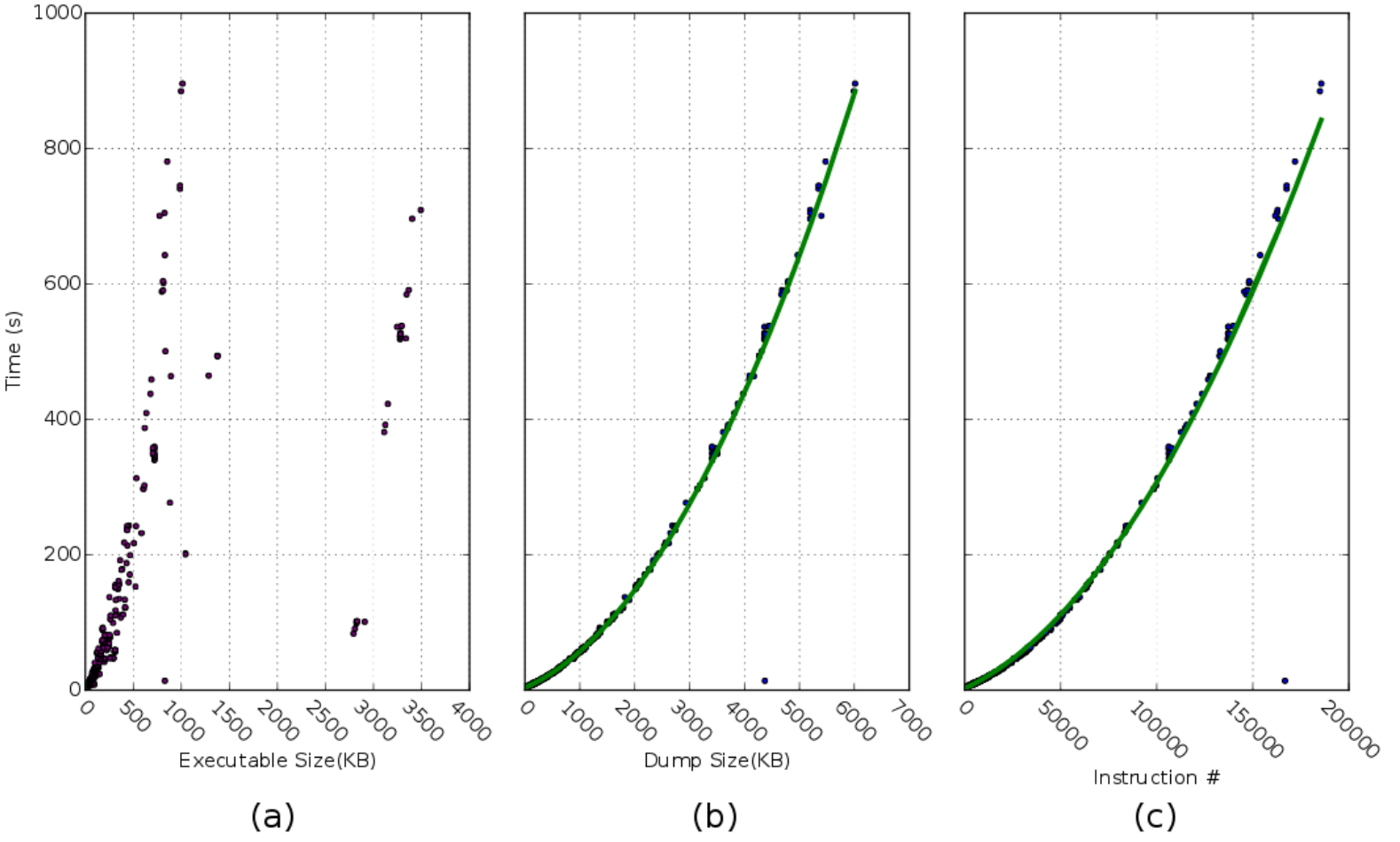
LISC Evaluation
Executable dumping







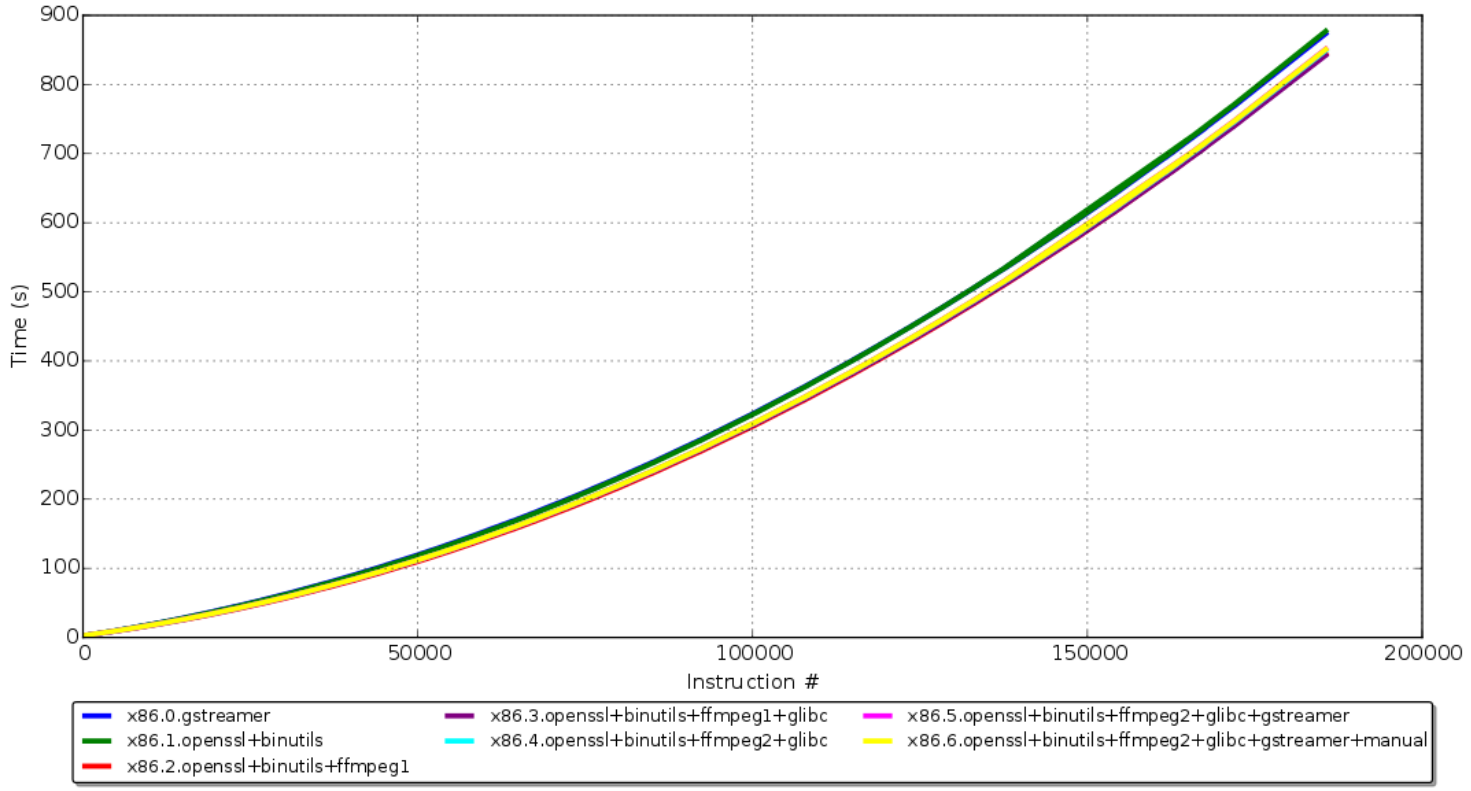
LISC Evaluation
Lifting Time Growth



LISC Evaluation
Performance Evaluation

- The authors argued linear growth, while I obtained a clear quadratic growth
  - Around 13 hours to lift around 900
- Not all executables could be lifted in a feasible time
- *Ldconfig* is the sole outlier
  - Also had the absolute worst lifting results
  - Although it has some particularities, like highest number of nops, no pattern could be found in its dump to explain this fact

LISC Evaluation
Automata Impact on Performance



LISC Evaluation
Automata Impact on Performance

- Another performance question not explored by the authors
- Bigger automatas in some cases actually *decrease* the lifting time
- Overall, all the results were very similar

LISC Evaluation Conclusions

- The framework and instructions provided by the authors were not sufficient for complete reproduction of results
- Several discrepancies between results reported in the paper and my own
- Performance experiments reported are very incomplete
  - Focused on the aspect that is the least burden to time

THANK YOU!