Accelerating Legacy String Kernels via Bounded Automata Learning

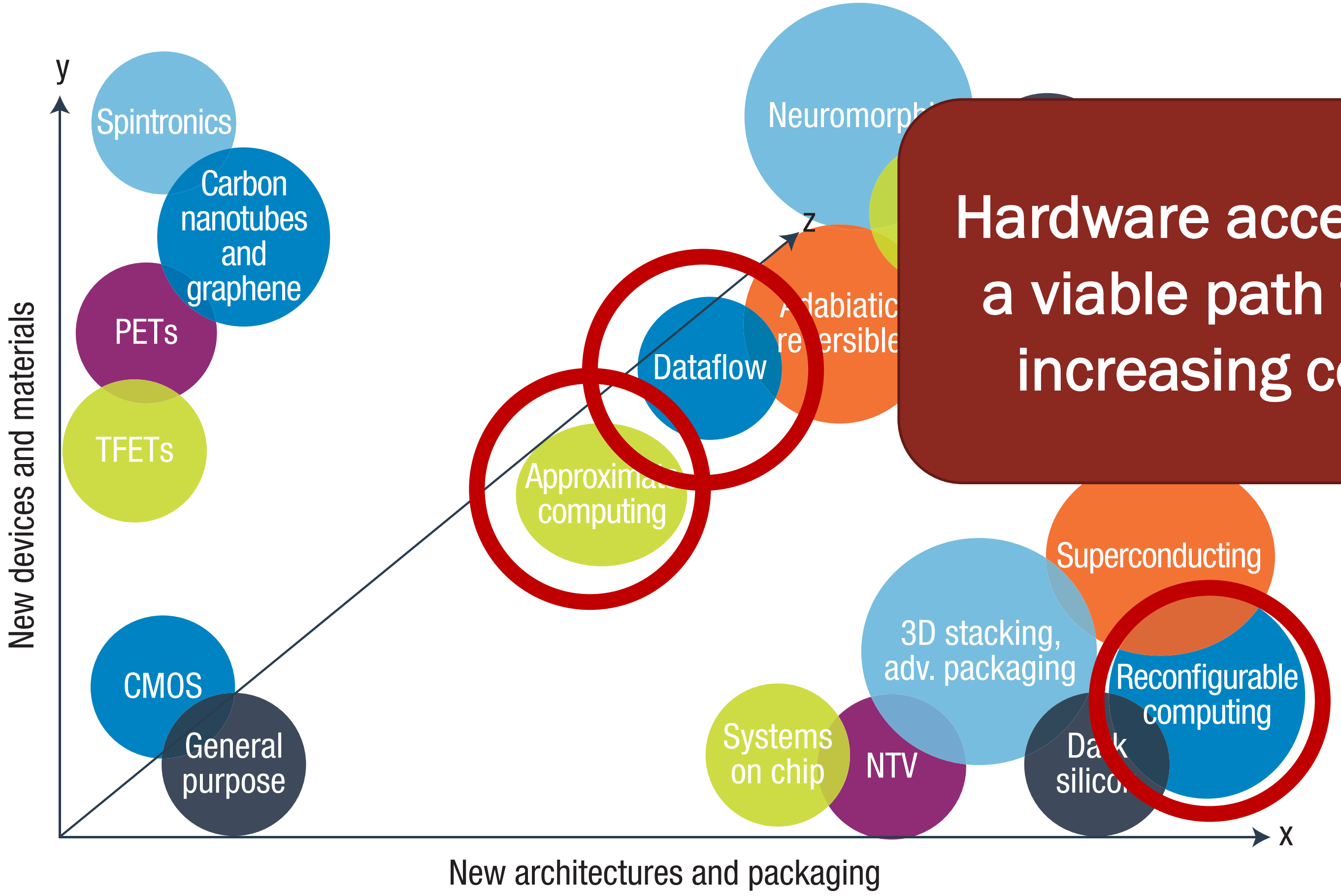
Kevin Angstadt†, Jean-Baptiste Jeannin‡, Westley Weimer†

†Computer Science and Engineering, University of Michigan
‡Aerospace Engineering, University of Michigan

COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

This work is funded in part by: NSF grants CCF-1629450, CCF-1763674, CCF-1908633; AFRL Contract No. FA8750-19- 1-0501; and the Jefferson Scholars Foundation.

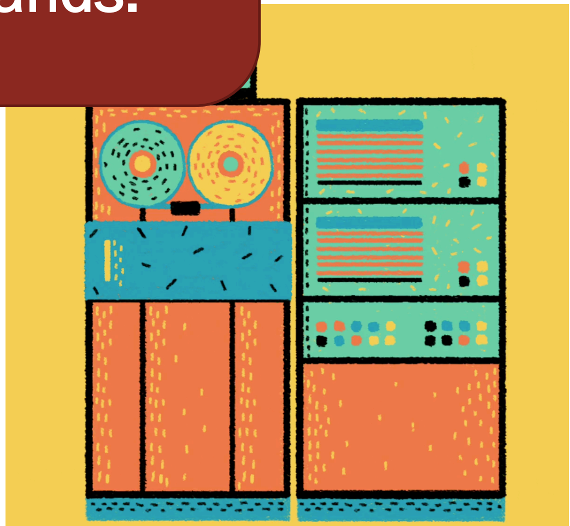
Physical Limits Spark Creativity



Hardware accelerators are seen as a viable path forward for tackling increasing compute demands.

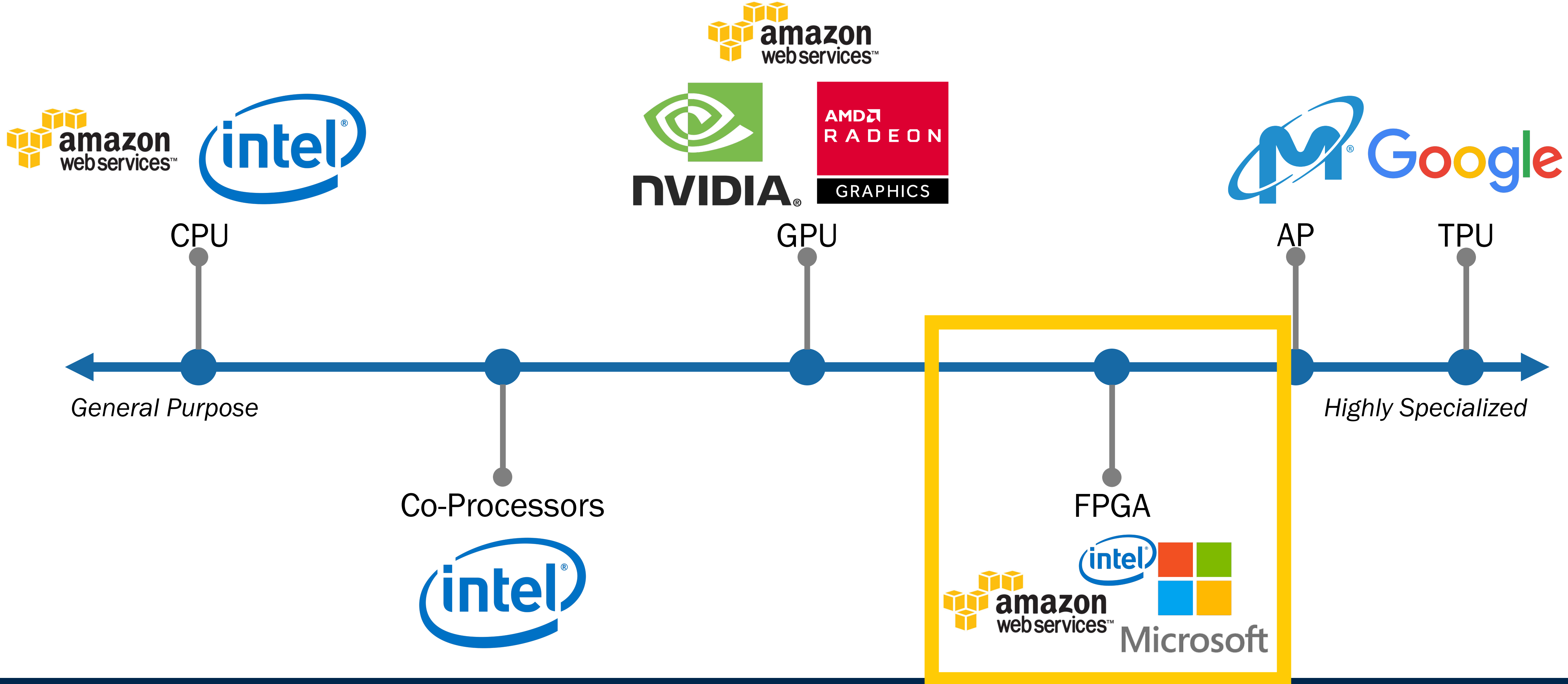
J. M. Shalf and R. Leland, "Computing Beyond Moore's Law". IEEE Computer, 2015.

Source: Dazeinfo



Source: MIT Technology Review

New Kinds of Processors



New Kinds of Processors

FORTUNE

POINTCLOUD

Official At Last: Intel Completes $1
Billion Buy of Altera

Why Microsoft Has Bet on FPGAs to Infuse Its Cloud With AI

ything from

Xilinx provides Alibaba Cloud FaaS with AI Acceleration

by NICK FARRELL on 10 MARCH 2020

AWS News Blog

EC2 F1 Instances with FPGAs – Now Generally Available

by Jeff Barr | on 19 APR 2017 | in Amazon EC2 | Permalink | Share

Legacy Code in the Age of Hardware Accelerators

- Legacy code typically cannot be directly compiled for FPGAs
- Learning a new programming model is costly and slows rate of adoption of new accelerators
- May want to “try out” new hardware with existing software
  - No training on new hardware
  - Limited time or resources to allocate

Talk Overview

- Background and Motivation
- Technical Approach
  - High-Level Summary
  - Problem Statement
  - Approach Details
  - Formal Results
- Empirical Evaluation
- Open Challenges

Goal: Aid developers programming FPGAs by automatically porting certain classes of existing source code without requiring low-level hardware knowledge to produce performant code

AutomataSynth at a Glance

- Framework for executing code (legacy software) on FPGAs and other hardware accelerators
- Dynamically observe and statically analyze program behavior to synthesize a functionally-equivalent hardware design
- Initial effort infers a set of finite automata rather than attempting to directly compile code
- Novel combination of model learning (learning theory), software model checking (software engineering), string decision procedures (PL/theory), and high-performance automata architectures (hardware)

Why Automata(Synth)?

- FPGA designs are often described in terms of state machines
- Automata a versatile and broadly-applicable
- Can build on significant research effort for accelerating state machine execution
- Other high-level approaches (cf. HLS) generally fail to abstract low-level architectural details
- Our approach decouples high-level program and low-level implementation

Automata Accelerate Big Data Applications

Detecting Intrusion Attempts in Network Packets

Learning Association Rules with an *a priori* approach

Detecting incorrect POS tags in NLP

Looking for Virus Signatures in Binary Data

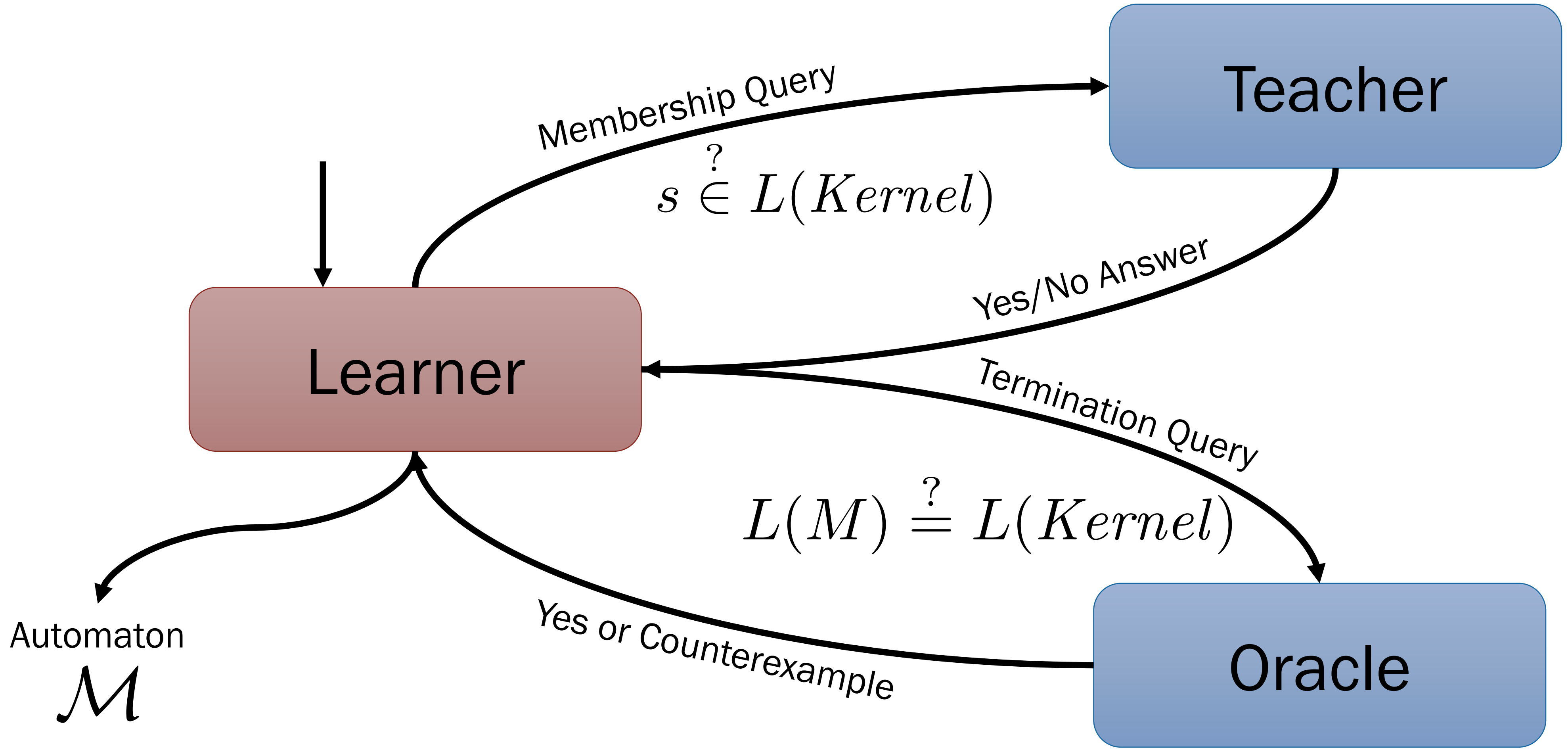
Detecting Higgs Events in Particle Collider Data

Aligning DNA Fragments to the Human Genome

Problem Statement (First Efforts)

- Input: function `kernel : string -> bool`
- Assumptions:
  - Function decides a regular language
  - Source code for function is available
- Output: finite automaton with the same behavior on “all” inputs as kernel

Angluin-Style Learning (L*)



Membership Queries are Direct

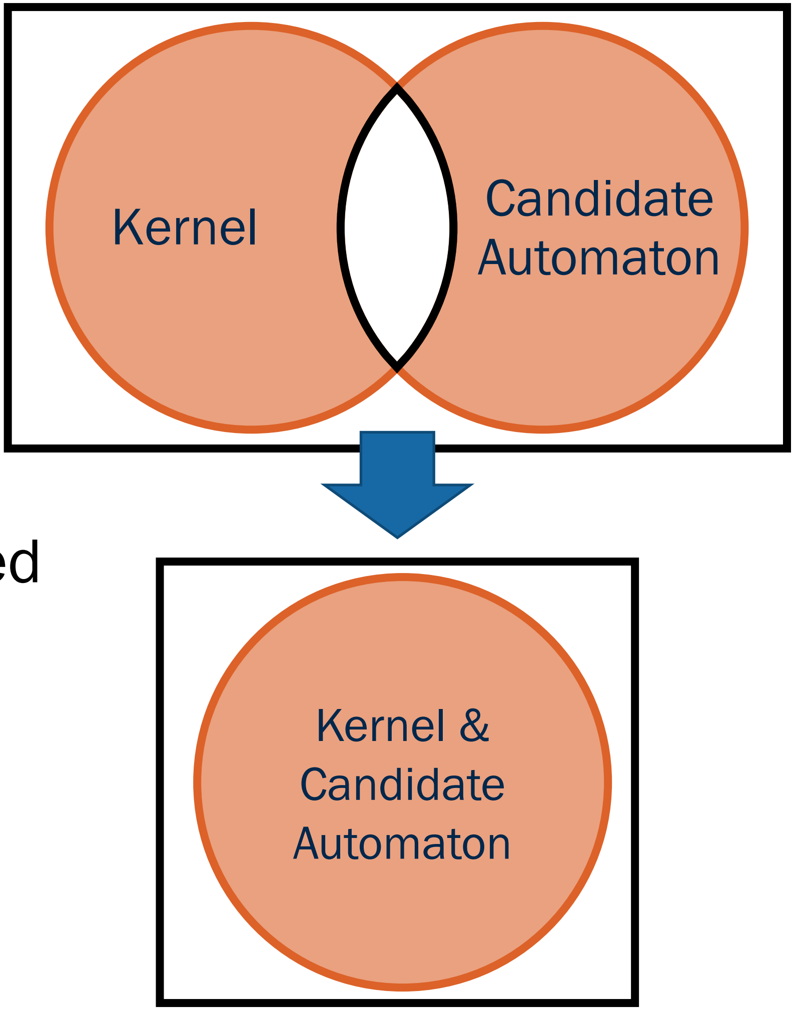
$$s \overset{?}{\in} L(Kernel)$$

- Check if kernel accepts input by running the code
- Return value of the kernel is the answer from the teacher
- **Caution:** take care with ASCII encoding and null terminators (not all functions assume C-style strings)

Understanding Termination Queries

$$L(M) \stackrel{?}{=} L(Kernel)$$

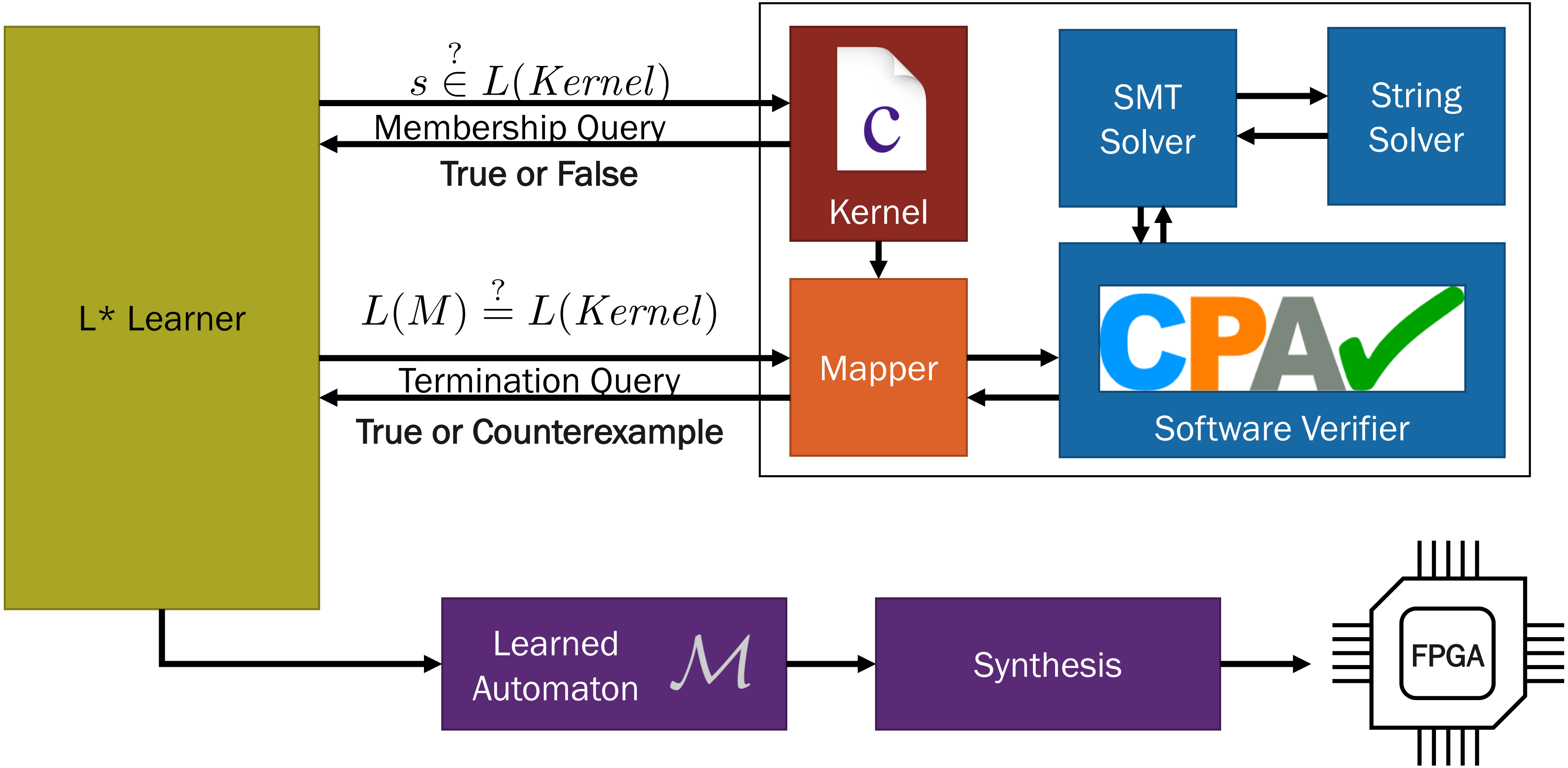
- Don’t have held-out automaton for comparison
- Test inputs generally do not suffice
  - Coverage, generation, etc. difficult challenges
- Constraint over string inputs
  - No inputs that are accepted by the kernel are rejected by the candidate machine (and vice versa)
  - “The symmetric difference is empty”
  - Allows for formulation as a software verification query



Equality Checking as Software Verification

- Explores control flow graph looking for property violations
  - Success finding variety of bugs (e.g., double-free, locking violations, etc.)
  - Used in industry for driver verification
- Bounded Model Checking suitable for this domain
  - Verifies that property holds for all program executions up to length k (i.e., fixed number of loop unrollings)
  - Incremental unrolling to check longer and longer executions
  - Use theorem prover to identify executions that violate property
- Wrapper program to encode the “symmetric difference” property
- Add in string solver to generate counterexamples

AutomataSynth System Architecture



Caveats/Challenges

Theorem provers are relatively complete.

- Software verification will occasionally return an unknown result
- No counterexample is produced, so L* cannot continue
- **Implication:** resulting automaton is approximate, but correct for all inputs shorter than some fixed bound

BMC with incremental unrolling is a semi-algorithm.

- Unrolling of program with infinite loops could continue indefinitely
- Termination query might never terminate
- For regular languages finite unrolling suffices (See §4.3)
- **Implication:** BMC+string solver will terminate and satisfies requirements for Termination Queries

Theoretical Implications

- When AutomataSynth terminates, we report if the automaton is correct or approximate
  - Formal approach means that automata are provably correct
  - Approximate automata are correct for inputs up to a known bound
- For functions deciding a regular language, correctness is guaranteed (modulo the theorem prover)
- In practice, we make use of timeouts to terminate AutomataSynth
  - Tunable to help define bounds of correctness

Evaluation: Guiding Research Questions

- How many real-world string kernels can AutomataSynth correctly learn? With approximation?

- Does AutomataSynth learn automata that fit within the design constraints of modern, automata-derived, reconfigurable architectures?

Experimental Methodology

- Mine GitHub for string functions in top C repositories
- Use Cil framework to iteratively parse each source file an extract all string functions
- Filter for duplicates and manual analysis to filter on Boolean return type
- Considered 26 repositories, 973 separate string functions, **18 meaningfully-distinct real-world benchmarks**
  - AutomataSynth did not support 3 due to functionality of underlying string solver (e.g., no math on characters)

Empirical Results

Benchmark	Project	LOC	Member Queries	Term. Queries	States	Runtime (min)	Correct
git_offset_1st_component	Git: Revision control	6	4,090	2	2	0.12	✓
checkerrormsg	jq: Command-line JSON processor	4	32,664	2	15	1436.58	✓*
checkfail		14	189,013	3	35	1438.47	✓*
skipline		17	7,663	3	3	4.90	✓
end_line	Linux: OS kernel	11	510,623	4	44	491.88	✓
start_line		11	206,613	2	46	80.22	Approx.
is_mcounted_section_name		54	672,041	7	57	1439.98	Approx.
is_numeric_index	MASSCAN: IP port scanner	17	10,727	3	4	4.95	✓
is_comment		11	4,090	2	2	0.23	✓
AMF_DecodeBoolean	OBS Studio: Live streaming and recording software	2	2,557	2	2	0.07	✓
cf_is_comment		28	4,599	2	4	5.00	✓
cf_is_splice		22	1,913	2	4	0.05	✓
is_reserved_name		39	240,705	8	42	1424.48	✓
has_start_code		18	10,213	2	7	0.08	✓
stbtt__isfont	Openpilot: Open-source driving agent	24	79,598	5	19	0.22	✓

Empirical Results

Benchmark	Project	LOC	Member	Term.		Runtime (min)	Correct
`git_offset_1st_component`	Git: Revision control					0.12	✓
`checkerrormsg`	jq: Command-line JSON processor					1436.58	✓*
`checkfail`						1438.47	✓*
`skipline`						4.90	✓
`end_line`	Linux: OS kernel					491.88	✓
`start_line`						80.22	Approx.
`is_mcounted_section_name`		54	672,041	7	57	1439.98	Approx.
`is_numeric_index`	MASSCAN: IP port scanner	17	10,727	3	4	4.95	✓
`is_comment`		11	4,090	2	2	0.23	✓
`AMF_DecodeBoolean`	OBS Studio: Live streaming and recording software	2	2,557	2	2	0.07	✓
`cf_is_comment`		28	4,599	2	4	5.00	✓
`cf_is_splice`		22	1,913	2	4	0.05	✓
`is_reserved_name`		39	240,705	8	42	1424.48	✓
`has_start_code`		18	10,213	2	7	0.08	✓
`stbtt__isfont`	Openpilot: Open-source driving agent	24	79,598	5	19	0.22	✓

AutomataSynth learns 13/18 kernels correctly and a further 2 approximately

Empirical Results

Benchmark	Project	LOC	Member Queries	Term. Queries	States	Runtime (min)	Correct
git_offset_1st_component	Git: Revision control	6	4,090	2	2	0.12	✓
checkerrormsg	jq: Command-line JSON processor	4	32,664	2	15	1436.58	✓*
checkfail		14	189,013	3	35	1438.47	✓*
skipline		17	7,663	3	3	4.90	✓
end_line	Linux: OS ke	11	510,623	4	44	491.88	✓
start_line						80.22	Approx.
is_mcounted_section_name						1439.98	Approx.
is_numeric_index	MASSCAN: IP scanner					4.95	✓
is_comment						0.23	✓
AMF_DecodeBoolean	OBS Studio: streaming a recording softwa					0.07	✓
cf_is_comment						5.00	✓
cf_is_splice						0.05	✓
is_reserved_name		39	240,705	8	42	1424.48	✓
has_start_code		18	10,213	2	7	0.08	✓
stbtt__isfont	Openpilot: Open-source driving agent	24	79,598	5	19	0.22	✓

Learning took an average of 7 hours. More than half take fewer than 5 minutes

Empirical Results

Benchmark	Project	LOC	Member Queries	Term. Queries	States	untime (min)	Correct
git_offset_1st_component	Git: Revision control	6	4,090	2	2	0.12	✓
checkerrormsg	jq: Command-line JSON processor	4	32,664	2	15	1436.58	✓*
checkfail		14	189,013	3	35	1438.47	✓*
skipline		17	7,663	3	3	4.90	✓
end_line	Linux: OS kernel	11	510,623	4	44	491.88	✓
start_line		11	206,613	2	46	80.22	Approx.
is_mcounted_section_name		54	672,041	7	57	1439.98	Approx.
is_numeric_index	MASSCAN: IP port	17	10,727	3	4	4.95	✓
is_comment					2	0.23	✓
AMF_DecodeBoolean					2	0.07	✓
cf_is_comment					4	5.00	✓
cf_is_splice					4	0.05	✓
is_reserved_name	r				42	1424.48	✓
has_start_code					7	0.08	✓
stbtt__isfont	sou				19	0.22	✓

Learned automata fall within resource constraints of FPGA-based architectures

Open Challenges: Guiding Desires

- Support broader classes of functions
  - Not all legacy code consists solely of Boolean string kernels
- More indicative benchmark applications
  - Mined real-world functions are a start, but…
  - Functions that dominate runtime
  - Larger or more complex functions
- Comparison of learning-based solutions (AutomataSynth) with HLS

Open Challenges (Learning-Specific)

1. Learn More Expressive Models (Transducers, Pushdown, etc.)
2. Improve String Solvers (Expressiveness, Performance, etc.)
3. Scale Termination Queries and Explore Alternatives
4. Understand Approximation (PAC Learning, Measure Error)

AutomataSynth Summary

- Framework for accelerating legacy Boolean string kernel functions using FPGAs
- Static and dynamic analyses of program behavior to construct functionally-equivalent automata
- Novel combination of Angluin-style learning with software model checking and string solvers
- Successfully constructs equivalent (or near equivalent) FPGA designs for more than 80% of real-world benchmarks mined from GitHub
- Many open challenges mean many opportunities for studying learning-based approaches for porting code

Source code: https://github.com/kevinaangstadt/automata-synth