Abstraction As A Practical Debugging Tool

Sandip Ray
Department of Computer Sciences
University of Texas at Austin
sandip@cs.utexas.edu
http://www.cs.utexas.edu/∼sandip

Broad Motivation

Dynamic and formal verification techniques have orthogonal strengths.

Broad Motivation

Dynamic and formal verification techniques have orthogonal strengths.

Dynamic:

- ✓ Easy applicability on practically any design
- ✓ Fast detection of easy-to-find bugs
- × Effectiveness decreases as shallow bugs are exhausted

Broad Motivation

Dynamic and formal verification techniques have orthogonal strengths.

Dynamic:

- ✓ Easy applicability on practically any design
- ✓ Fast detection of easy-to-find bugs
- × Effectiveness decreases as shallow bugs are exhausted

Formal:

- ✓ Effective for finding difficult corner cases
- × Too expensive to employ when there are too many shallow bugs

Broad Motivation

Dynamic and formal verification techniques have orthogonal strengths.

Dynamic:

- ✓ Easy applicability on practically any design
- ✓ Fast detection of easy-to-find bugs
- × Effectiveness decreases as shallow bugs are exhausted

Formal:

- ✓ Effective for finding difficult corner cases
- × Too expensive to employ when there are too many shallow bugs

In current industrial practice, formal is quite isolated from traditional functional verification tool flows.

How can we achieve closer and synergistic integration of these two complementary techniques?

Project Goals

Objective: Use formal analysis techniques to help simulation-based verification find deep corner-case bugs.

Project Goals

Objective: Use formal analysis techniques to help simulation-based verification find deep corner-case bugs.

Rationale: Simulation is limited by the difficulty of producing appropriate stimuli to target deep corner cases. Formal methods can help.

Project Goals

Objective: Use formal analysis techniques to help simulation-based verification find deep corner-case bugs.

Rationale: Simulation is limited by the difficulty of producing appropriate stimuli to target deep corner cases. Formal methods can help.

Idea: Make use of light-weight abstraction generation to construct simpler models that can be effectively simulated.

Project Goals

Objective: Use formal analysis techniques to help simulation-based verification find deep corner-case bugs.

Rationale: Simulation is limited by the difficulty of producing appropriate stimuli to target deep corner cases. Formal methods can help.

Idea: Make use of light-weight abstraction generation to construct simpler models that can be effectively simulated.

Focus: RTL implementations of pipelines and hardware communication protocols.

Summary

A tool for generating a range of abstractions of an RTL design, that preserves a specific given observation.

Summary

A tool for generating a range of abstractions of an RTL design, that preserves a specific given observation.

- Each abstraction is conservative with respect to the observation.
- The tool is parameterized by the amount of abstraction desired.
- Abstraction parameters are user-controlled to target various design facets.

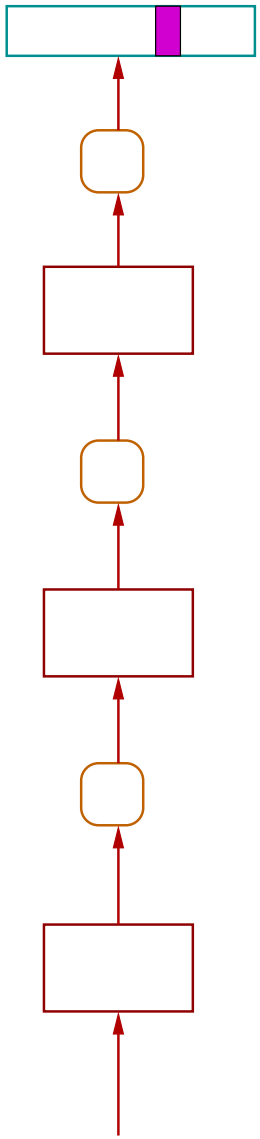
Summary

A tool for generating a range of abstractions of an RTL design, that preserves a specific given observation.

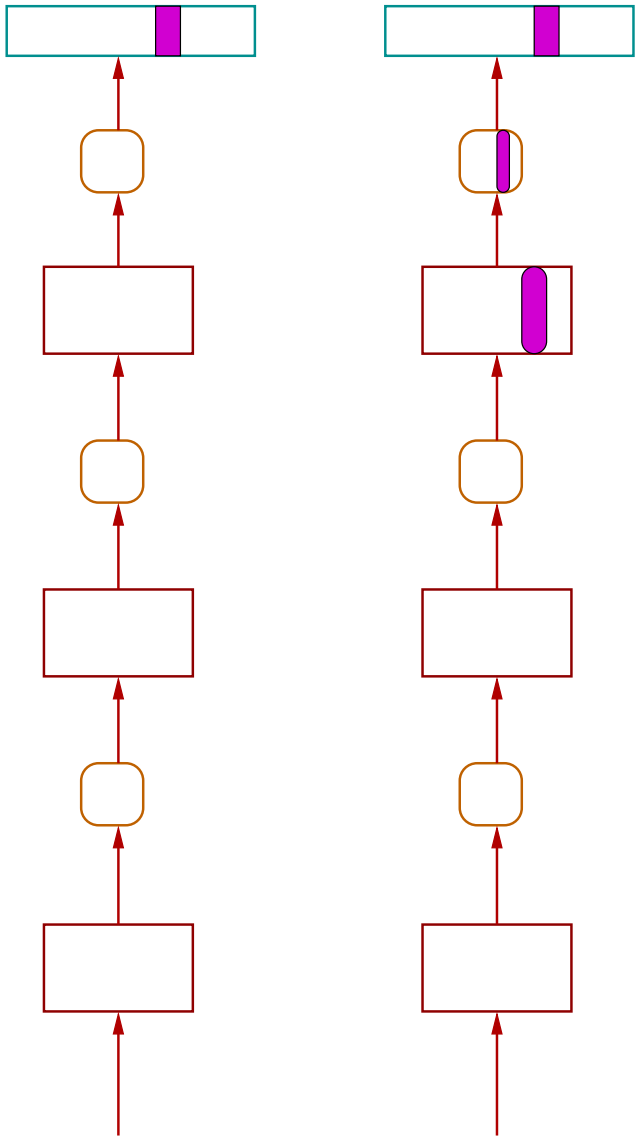
- Each abstraction is conservative with respect to the observation.
- The tool is parameterized by the amount of abstraction desired.
- Abstraction parameters are user-controlled to target various design facets.

Tool used to abstract RTL implementations of pipelined transaction queues.

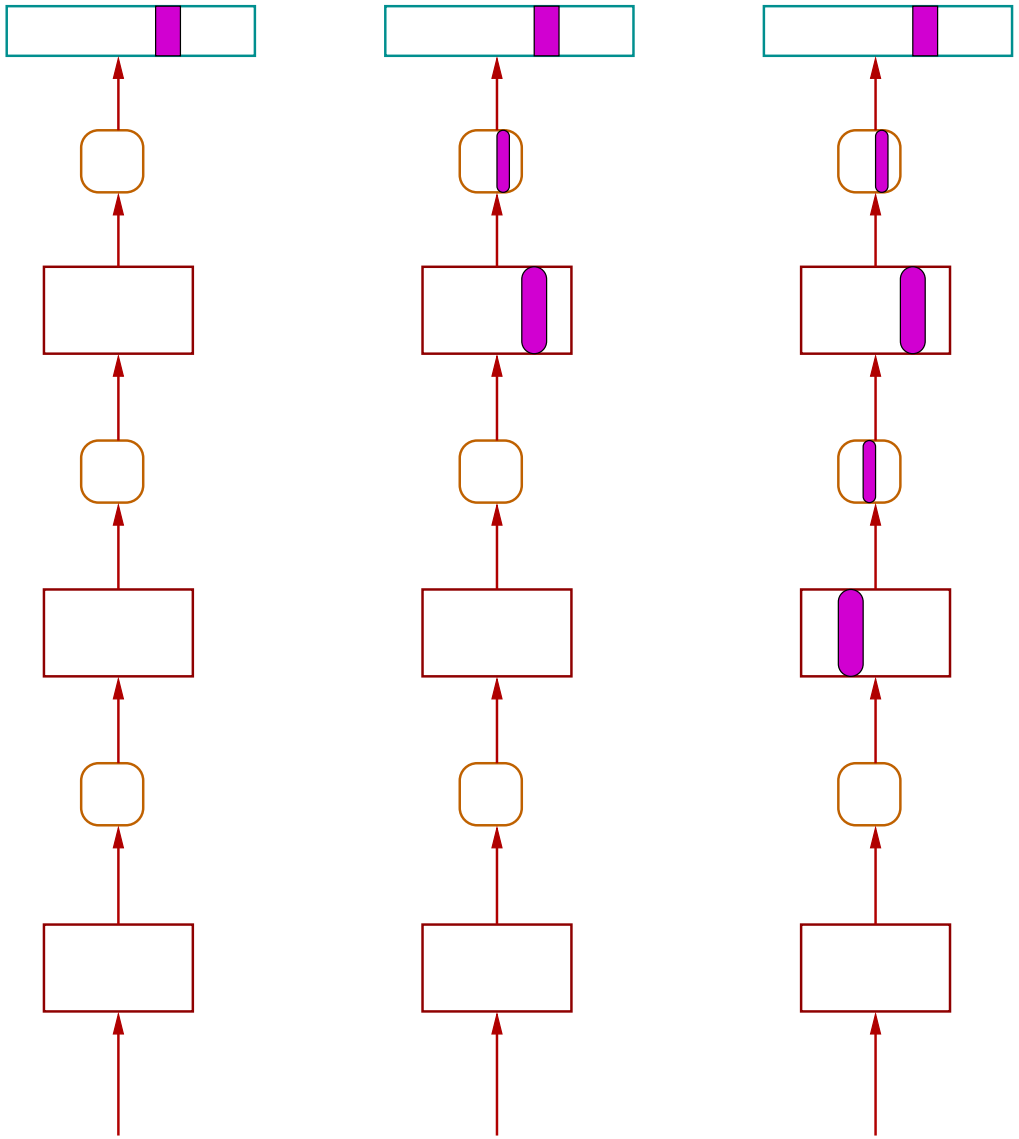
How Do We Generate Abstractions?



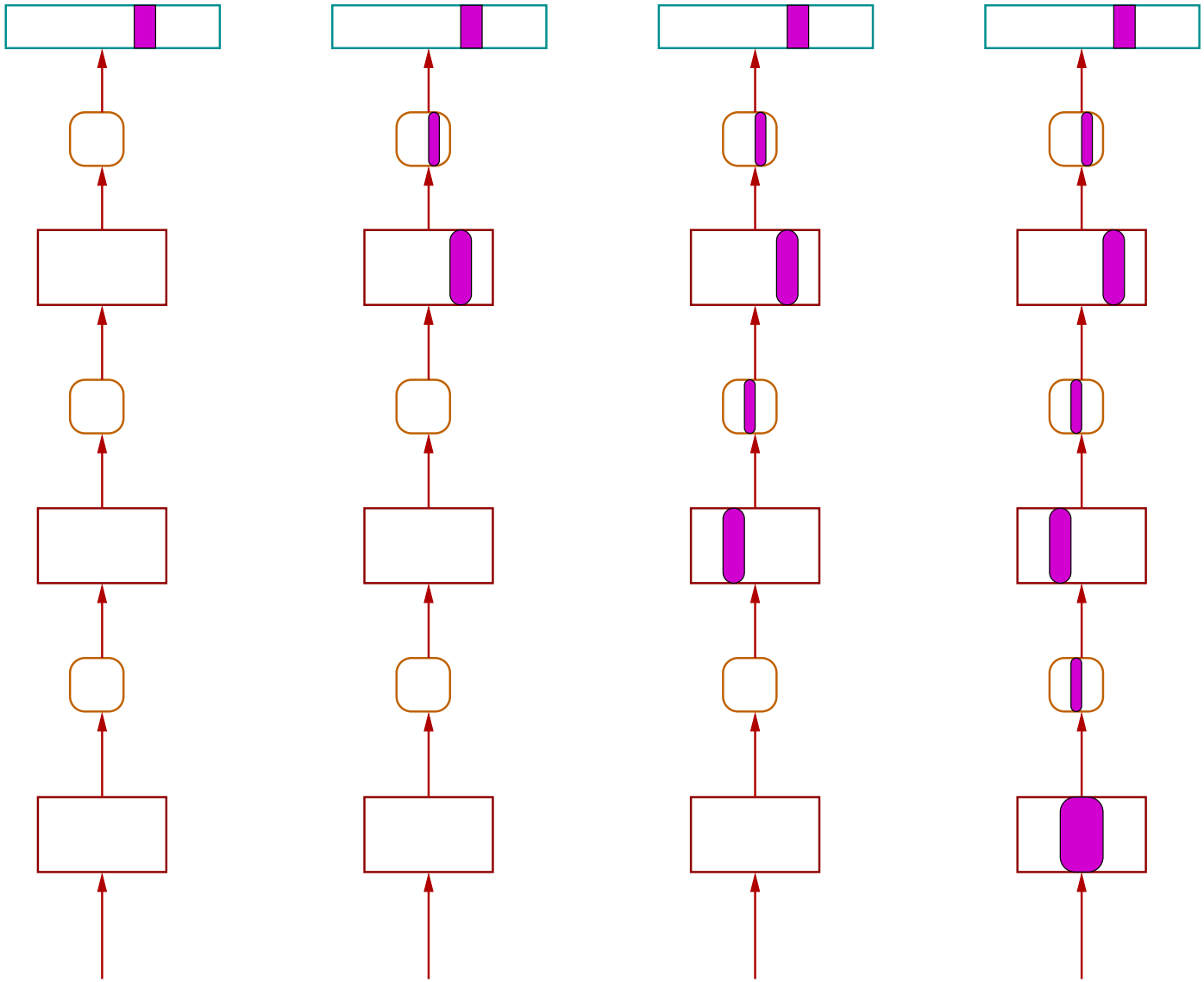
How Do We Generate Abstractions?



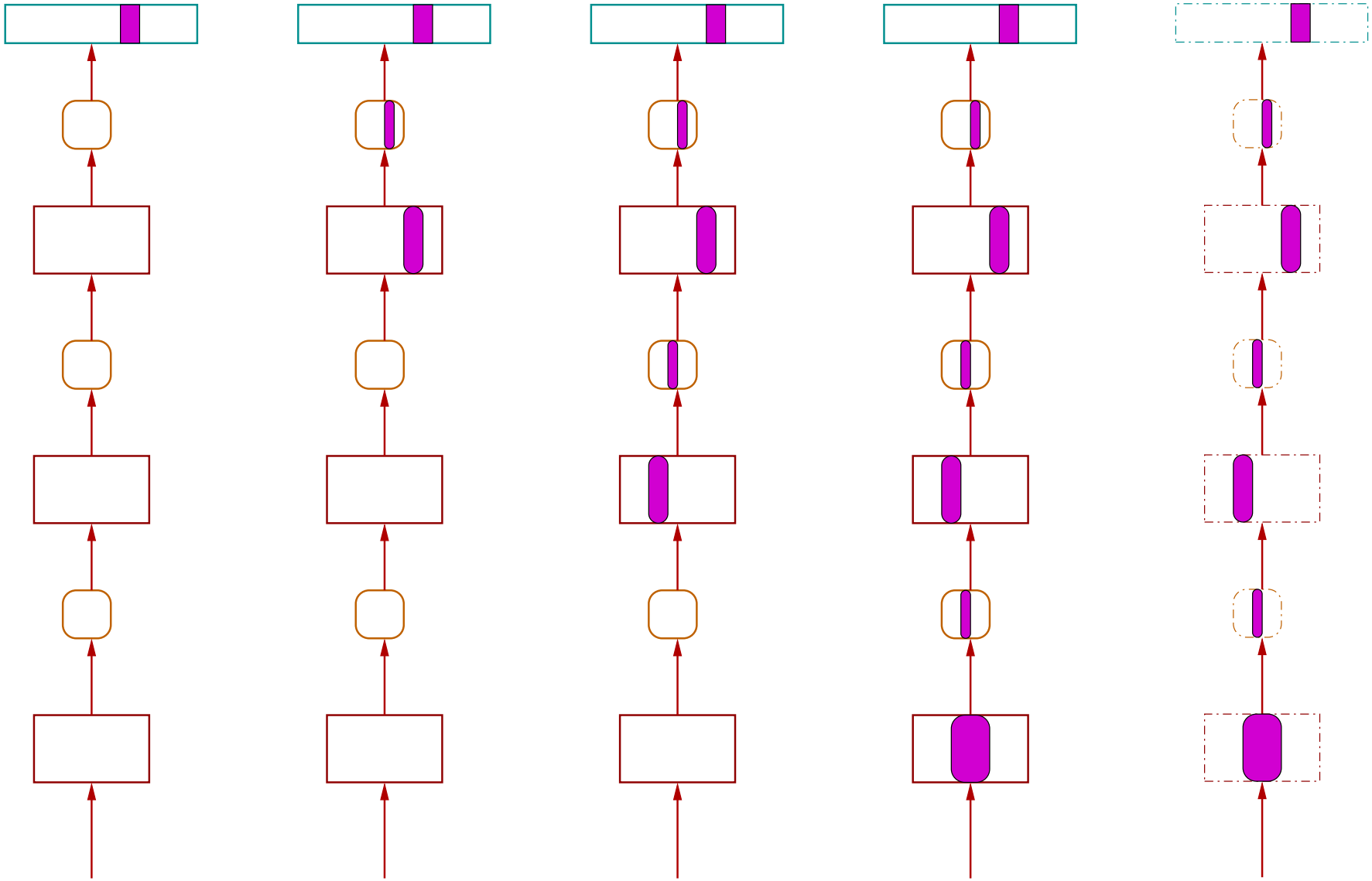
How Do We Generate Abstractions?



How Do We Generate Abstractions?



How Do We Generate Abstractions?



But Wait!

Isn't this just cone-of-influence reduction?

But Wait!

Isn't this just cone-of-influence reduction?

Yes, in principle, but with some important differences.

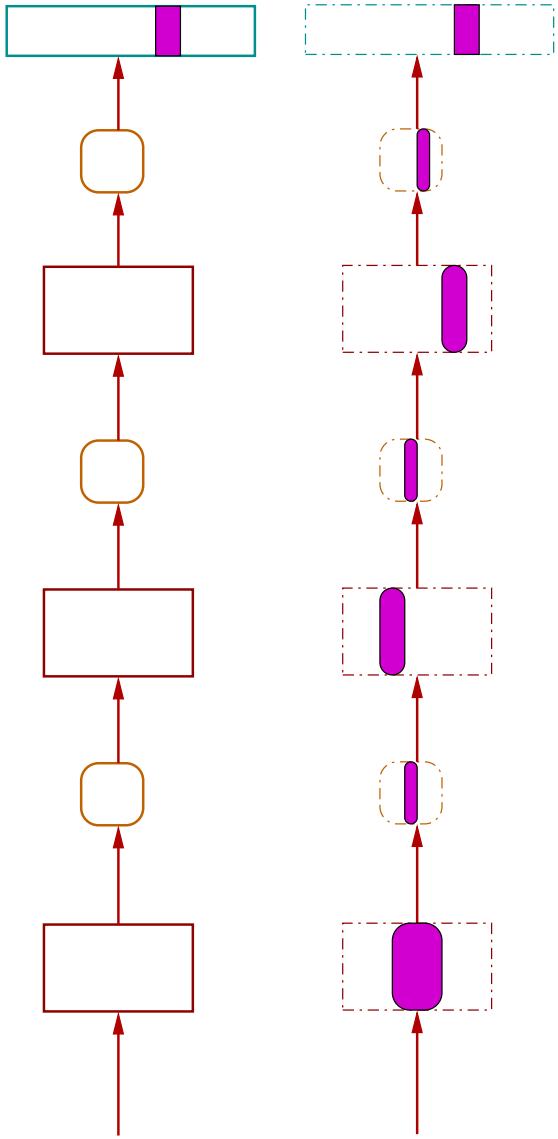
But Wait!

Isn't this just cone-of-influence reduction?

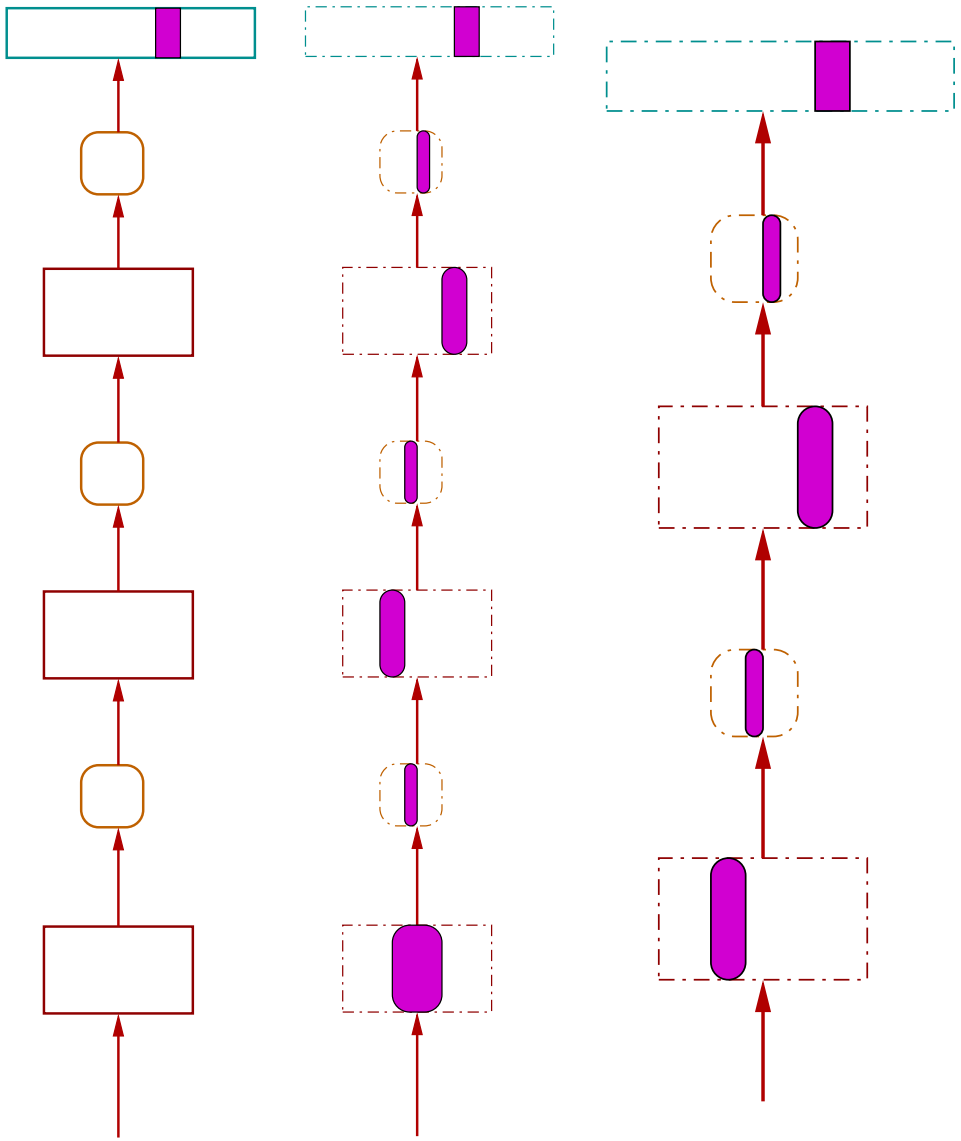
Yes, in principle, but with some important differences.

- These are not just syntactic checks of variables. We must comprehend the semantics of operations.
- Since the goal is to help simulation, not target formal correctness, the abstraction generation requirements are different.

How Much Do We abstract?



How Much Do We abstract?



The answer is driven by which stage simulation wishes to target.

Similar questions arise in dealing with non-determinism.

It is critical for the tool to be configurable for generating a range of abstractions.

Efficiency of Model Generation

It is critical that we generate the models very fast.

Efficiency of Model Generation

It is critical that we generate the models very fast.

- After all, we are trying to generate a large number of abstractions, and many of them will be spurious.

Efficiency of Model Generation

It is critical that we generate the models very fast.

- After all, we are trying to generate a large number of abstractions, and many of them will be spurious.

Our current procedure works well in practice.

- Generated over 500 abstractions of OR1200 instruction pipeline in a few seconds.

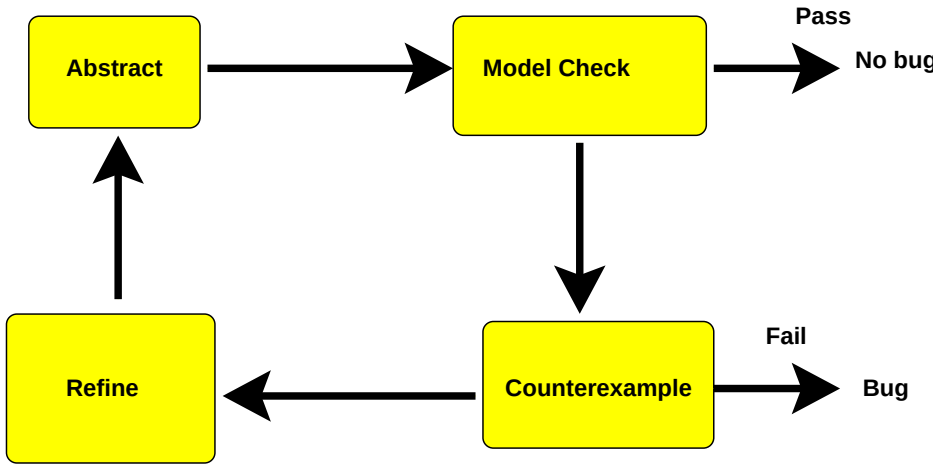
We are currently looking to apply the approach on cache coherence and other protocols.

Difference with Traditional Abstractions

Abstraction is the bread-and-butter of formal verification techniques.

Difference with Traditional Abstractions

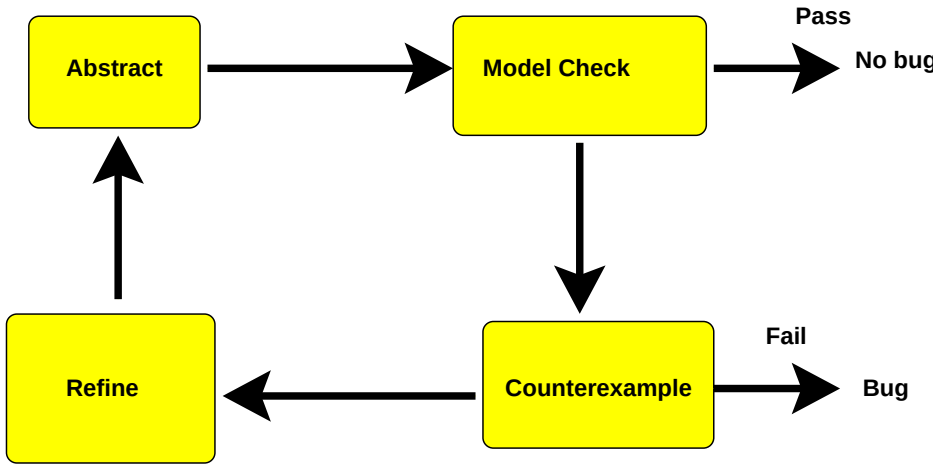
Abstraction is the bread-and-butter of formal verification techniques.



There has been significant work on effective generation and refinement of abstractions.

Difference with Traditional Abstractions

Abstraction is the bread-and-butter of formal verification techniques.

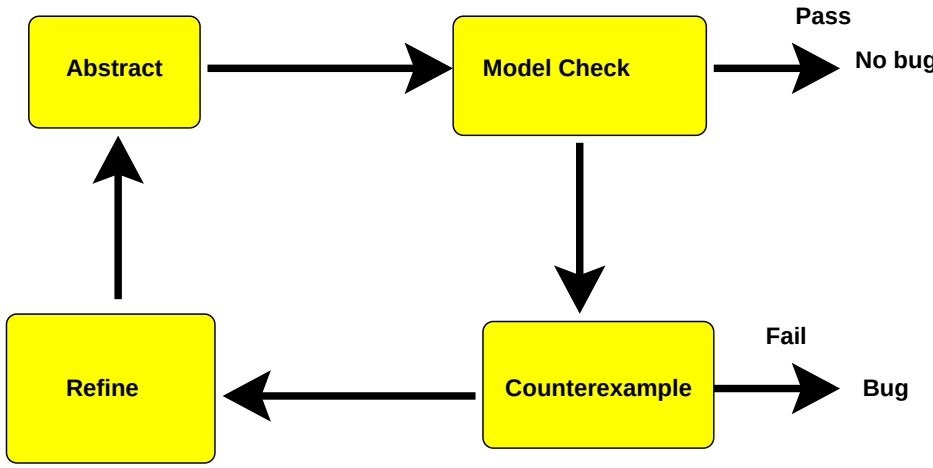


There has been significant work on effective generation and refinement of abstractions.

But the focus of these abstractions is on constructing a formal proof, not models for simulation.

Difference with Traditional Abstractions

Abstraction is the bread-and-butter of formal verification techniques.



There has been significant work on effective generation and refinement of abstractions.

But the focus of these abstractions is on constructing a formal proof, not models for simulation.

Our tool is a fast, configurable, but not necessarily complete.

Concluding Remarks

Simulation has long been used to assist formal analysis techniques, but relatively less work has been done in the other direction.

Concluding Remarks

Simulation has long been used to assist formal analysis techniques, but relatively less work has been done in the other direction.

- I believe it is time for formal to give something back!

Concluding Remarks

Simulation has long been used to assist formal analysis techniques, but relatively less work has been done in the other direction.

- I believe it is time for formal to give something back!

Application of **heavy-weight** formal methods is expensive, but it is possible to use it in light-weight manner to assist simulation.

Concluding Remarks

Simulation has long been used to assist formal analysis techniques, but relatively less work has been done in the other direction.

- I believe it is time for formal to give something back!

Application of **heavy-weight** formal methods is expensive, but it is possible to use it in light-weight manner to assist simulation.

I believe it is by such synergistic combination of formal and dynamic that we can reach the target of a scalable and efficient verification paradigm.

Questions?

Thank You!