



Partial Models and Software Model Checking

Marsha Chechik
 University of Toronto

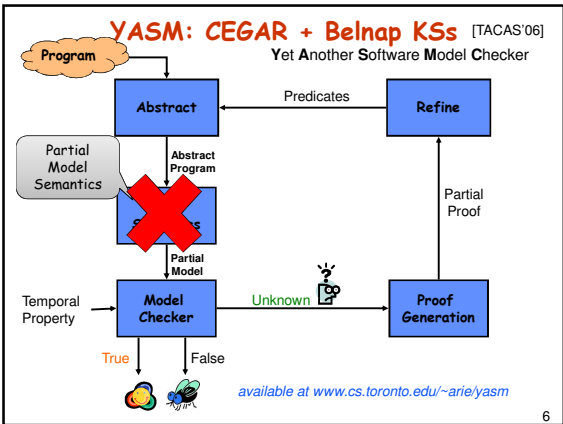
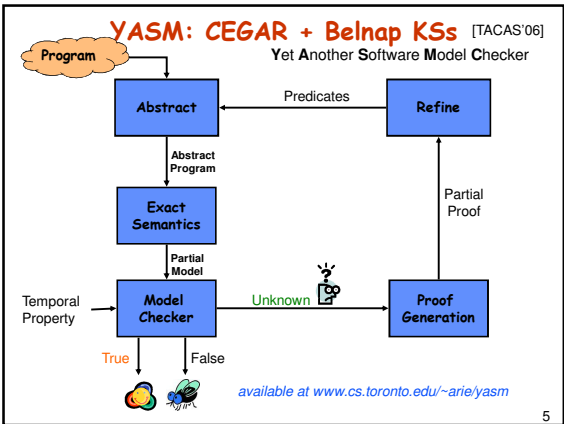
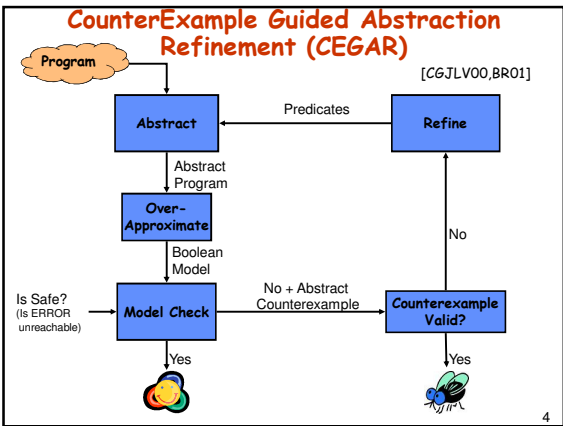
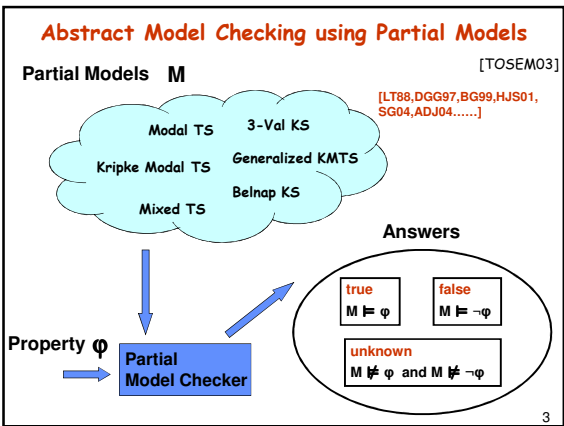
Arie Gurfinkel
 SEI/CMU

MLQA
 July 9, 2010

- ⇒ NO WARRANTY
- ⇒ THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.
- ⇒ Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.
- ⇒ This Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.
- ⇒ This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

2



Why use partial models in CEGAR?

- ⇒ Prove and disprove properties
- ⇒ Check complex properties
 - ↳ reachability, non-termination, existential (exists-a-path), universal (forall-paths), branching (CTL)
- ⇒ Use all information available from abstraction
 - ↳ traditional predicate abstraction already includes over- and under-approximation
- ⇒ Exploit new techniques and heuristics for efficient model-checking and refinement
 - ↳ refinement with 3-valued counterexamples
 - ↳ aggressive abstraction
 - ↳ error region refinement
 - ↳ and many more...

7

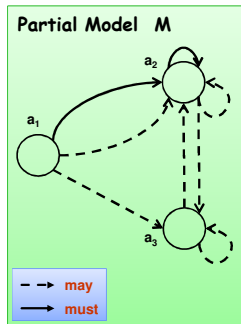
Outline

- ⇒ Partial Models for Program Abstraction
 - ↳ Overview of existing modeling formalisms
 - ↳ Program Abstraction "Problem"
 - ↳ Expressiveness of Partial Models
 - ↳ Model Checking w/ Reduced Inductive Semantics
- ⇒ Yasm: A Brief Overview
 - ↳ Models and Model Checking algorithm
 - ↳ Program Abstraction
 - ↳ Abstraction Refinement
 - ↳ Exploiting Partial Model Semantics
 - ↳ Checking Non-Termination of Recursive Programs
- ⇒ Conclusions and Lessons Learned

8

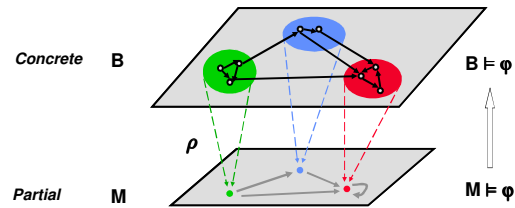
Partial Models

- ⇒ Transition system: states and transitions
- ⇒ Two types of transitions
 - ↳ *may*: possible (over-approximating) behaviors
 - ↳ *must*: necessary (under-approximating) behaviors



9

Refinement of Partial Models



Mixed Simulation [DGG97]

- B refines M (or M approximates B) iff there exists a relation ρ s.t.
 - B ρ -simulates necessary behaviors of M
 - Possible behaviors of M ρ -simulate B

10

Existing Partial Modeling Formalisms

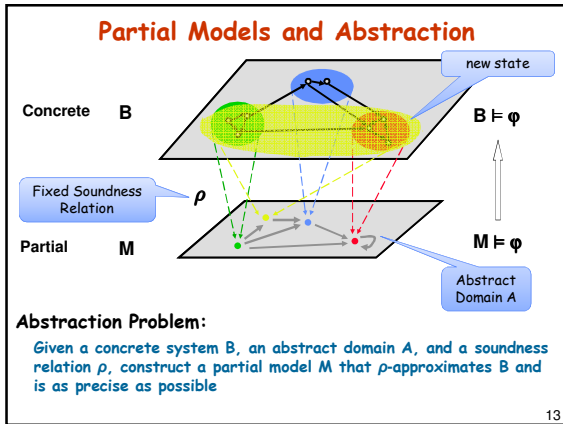
- ⇒ Kripke Modal Transition Systems [HJS01] **KMTSS**
 - ↳ *must* transitions \subseteq *may* transitions
 - ↳ a.k.a.: 3-valued Kripke Structures [CDEG03], Modal TSs [LT88], Partial Kripke Structures [BG99]
- ⇒ Mixed Transition Systems [DGG97] **MixTSs**
 - ↳ independent *must* and *may* transitions
 - ↳ a.k.a.: 4-valued (Belnap) Kripke Structures [CDEG03]
- ⇒ Generalized Kripke Modal Transition Systems [SG04] **GKMTSS**
 - ↳ with *must* hyper-transitions
 - ↳ a.k.a.: Abstract TSs [A6J04], Disjunctive Modal TSs [Larsen91]

11

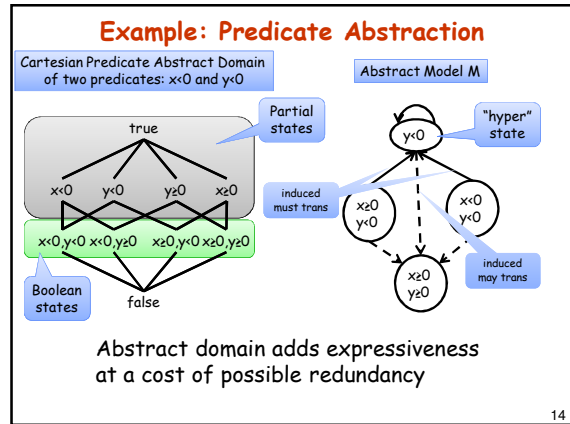
Existing Partial Modeling Formalisms

- ⇒ Kripke Modal Transition Systems **KMTSS**
 - ↳ *must* transitions
 - ↳ a.k.a.: 3-valued Modal TSs [LT88], Structures [BG99]
- ⇒ Mixed Transition Systems **MixTSs**
 - ↳ independent *mu* and *may* transitions
 - ↳ a.k.a.: 4-valued Structures [CDEG03]
- ⇒ Generalized Kripke Modal Transition Systems **GKMTSS**
 - ↳ with *mu* Hyper-Transitions
 - ↳ a.k.a.: Abstract TSs [A6J04], Disjunctive Modal TSs [Larsen91]

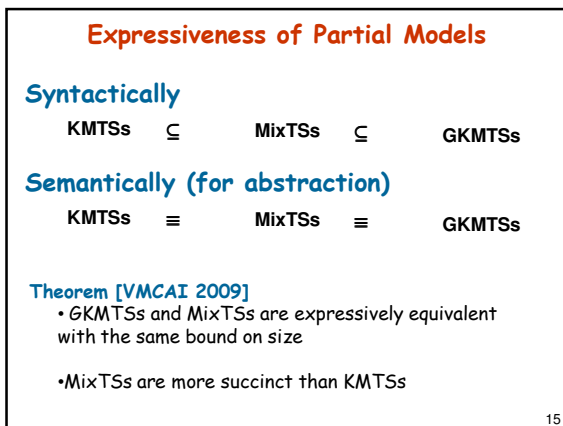
12



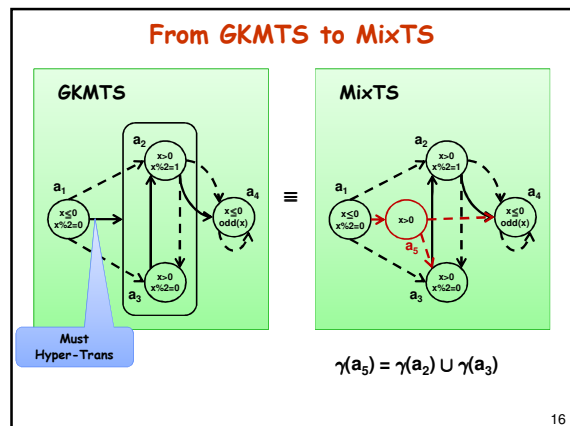
13



14



15



16

- ### Outline
- Partial Models for Program Abstraction
 - Overview of existing modeling formalisms
 - Program Abstraction "Problem"
 - Expressiveness of Partial Models
 - Model Checking w/ Reduced Inductive Semantics
 - Yasm: A Brief Overview
 - Models and Model Checking algorithm
 - Program Abstraction
 - Abstraction Refinement
 - Exploiting Exact Semantics
 - Checking Non-Termination of Recursive Programs
 - Conclusions and Lessons Learned

17

- ### Model Checking of Partial Models
- Fact:** Partial Modeling formalisms are not equivalent w.r.t Model Checking semantics!
 - Inductive semantics of temporal logic (SIS)**
 - defined inductively on the syntax of the logic, e.g., $\|\phi \wedge \psi\| \triangleq \|\phi\| \wedge \|\psi\|$, $\|\exists X\phi\| \triangleq \text{pre}(\|\phi\|)$
 - tractable, efficient, symbolic algorithm
 - Fact:** GKMTSs are strictly more precise than MixTSs w.r.t the Standard Inductive Semantics!
 - But...** ... hyper-transitions are hard to encode symbolically

18

SIS: MixTS is less precise than GKMTS

Property: $\exists x \forall y \neg q$

For model-checking under standard inductive semantics of TL, MixTS are less precise than GKMTS

Reduced Inductive Semantics

for each sub-formula ψ

1. evaluate ψ on boolean states only
2. extend to partial states

end for

Boolean states

Partial states

Theorem [VMCAI'09]

- ↳ RIS more precise than SIS
- ↳ GKMTS=MixTS under RIS

RIS for Predicate Abstraction

	TS + Over-Approx	MixTS + SIS	MixTS + RIS	GKMTS + SIS/RIS
Sets repr. (BDD vars)	N	2N + 1	N + 1	?
Trans repr. (BDD vars)	N + N	2N + 2N + 1	N + 2N + 1	?
Extra Ops	none	none	REDUCE	?
Precision	-	+	++	++
Efficiency	++	+	++	?

MixTS + RIS is a precise Partial Model w/ effective symbolic Model Checking procedure

RIS for Predicate Abstraction

to model true, false, unknown

# of predicates	TS + Over-Approx	MixTS + SIS	MixTS + RIS	GKMTS + SIS/RIS
Sets repr. (BDD vars)	N	2N + 1	N + 1	?
Trans repr. (BDD vars)	N + N	2N + 2N + 1	N + 2N + 1	?
Extra Ops	none	none	REDUCE	?
Precision	-	+	++	++
Efficiency	++	+	++	?

MixTS + RIS is a precise Partial Model w/ effective symbolic Model Checking procedure

Results

Model Size	n	SIS		RIS			
		Analysis (sec.)	a1 a2	Analysis (sec.)	REDUCTION (sec.)	a1	a2
Prop1	100	2.20	T U	3.60	0.74		
	200	15.36	T U	27.77	6.45	T	T
	250	28.92	T U	55.19	13.40	T	T
Prop2	100	3.60	T U	0.03	< 10 ⁻⁴		
	200	27.16	T U	0.12	< 10 ⁻⁴	T	T
	250	54.62	T U	0.19	< 10 ⁻⁴	T	T
Prop3	100	33.96	F F	21.24	4.5		
	200	395.24	F F	258.72	42.44	F	F
	250	1108.67	F F	546.88	101.20	F	F

- ↳ RIS is compact than SIS
- ↳ RIS is more precise than SIS
- ↳ RIS has the same complexity as SIS

Results

Model Size	n	SIS		RIS			
		Analysis (sec.)	a1 a2	Analysis (sec.)	REDUCTION (sec.)	a1	a2
Prop1	100	2.20	T U	3.60	0.74		
	200	15.36	T U	27.77	6.45	T	T
	250	28.92	T U	55.19	13.40	T	T
Prop2	100	3.60	T U	0.03	< 10 ⁻⁴		
	200	27.16	T U	0.12	< 10 ⁻⁴	T	T
	250	54.62	T U	0.19	< 10 ⁻⁴	T	T
Prop3	100	33.96	F F	21.24	4.5		
	200	395.24	F F	258.72	42.44	F	F
	250	1108.67	F F	546.88	101.20	F	F

RIS more compact

- ↳ RIS is compact than SIS
- ↳ RIS is more precise than SIS
- ↳ RIS has the same complexity as SIS

Results

		SIS				RIS			
Model Size	n	370,070				216,689			
	n	1,460,270				853,389			
	n	2,275,196				1,329,215			
Prop.	n	Analysis (sec.)	a ₁	a ₂	Analysis (sec.)	REDUCTION (sec.)	a ₁	a ₂	
Prop ₁	100	2.20			3.60	0.74			
	200	15.36	T	U	27.77	6.45	T	U	T
	250	28.92			55.19	13.40			
Prop ₂	100	3.60			0.03	< 10 ⁻⁴			
	200	27.16	T	U	0.12	< 10 ⁻⁴	T	U	T
	250	54.62			0.19	< 10 ⁻⁴			
Prop ₃	100	33.96			21.24	4.5			
	200	395.24	F	F	258.72	42.44	F	F	F
	250	1108.67			546.88	101.20			

↳ RIS is compact than SIS
 ↳ RIS is more precise than SIS
 ↳ RIS has the same complexity as SIS

RIS more precise

25

Results

		SIS				RIS			
Model Size	n	370,070				216,689			
	n	1,460,270				853,389			
	n	2,275,196				1,329,215			
Prop.	n	Analysis (sec.)	a ₁	a ₂	Analysis (sec.)	REDUCTION (sec.)	a ₁	a ₂	
Prop ₁	100	2.20			3.60	0.74			
	200	15.36	T	U	27.77	6.45	T	U	T
	250	28.92			55.19	13.40			
Prop ₂	100	3.60			0.03	< 10 ⁻⁴			
	200	27.16	T	U	0.12	< 10 ⁻⁴	T	U	T
	250	54.62			0.19	< 10 ⁻⁴			
Prop ₃	100	33.96			21.24	4.5			
	200	395.24	F	F	258.72	42.44	F	F	F
	250	1108.67			546.88	101.20			

↳ RIS is compact than SIS
 ↳ RIS is more precise than SIS
 ↳ RIS has the same complexity as SIS

Reduce ~ 25% of runtime

26

Results

		SIS				RIS			
Model Size	n	370,070				216,689			
	n	1,460,270				853,389			
	n	2,275,196				1,329,215			
Prop.	n	Analysis (sec.)	a ₁	a ₂	Analysis (sec.)	REDUCTION (sec.)	a ₁	a ₂	
Prop ₁	100	2.20			3.60	0.74			
	200	15.36	T	U	27.77	6.45	T	U	T
	250	28.92			55.19	13.40			
Prop ₂	100	3.60			0.03	< 10 ⁻⁴			
	200	27.16	T	U	0.12	< 10 ⁻⁴	T	U	T
	250	54.62			0.19	< 10 ⁻⁴			
Prop ₃	100	33.96			21.24	4.5			
	200	395.24	F	F	258.72	42.44	F	F	F
	250	1108.67			546.88	101.20			

↳ RIS is compact than SIS
 ↳ RIS is more precise than SIS
 ↳ RIS has the same complexity as SIS

Same Complexity

27

In Summary

- ↳ Using partial models
 - ↳ Use for abstraction is different than use for specification!
- ↳ Expressive power
 - ↳ The three main formalisms are expressively equivalent (when used for abstraction)
- ↳ Reasoning
 - ↳ Several model-checking semantics possible.
 - ↳ The choice dramatically affects precision!
- ↳ Verdict:
 - ↳ overall, MixTS + RIS makes a good precision/efficiency trade-off

28

Outline

- ↳ Partial Models for Program Abstraction
 - ↳ Overview of existing modeling formalisms
 - ↳ Program Abstraction "Problem"
 - ↳ Expressiveness of Partial Models
 - ↳ Model Checking w/ Reduced Inductive Semantics
- ↳ Yasm: A Brief Overview
 - ↳ Models and Model Checking algorithm
 - ↳ Program Abstraction
 - ↳ Abstraction Refinement
 - ↳ Exploiting Partial Model Semantics
 - ↳ Checking Non-Termination of Recursive Programs
- ↳ Conclusions and Lessons Learned

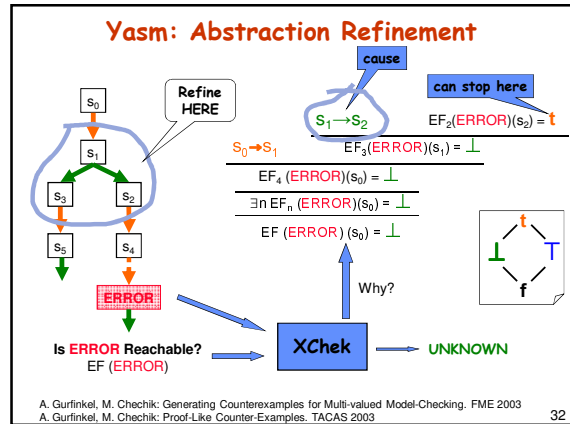
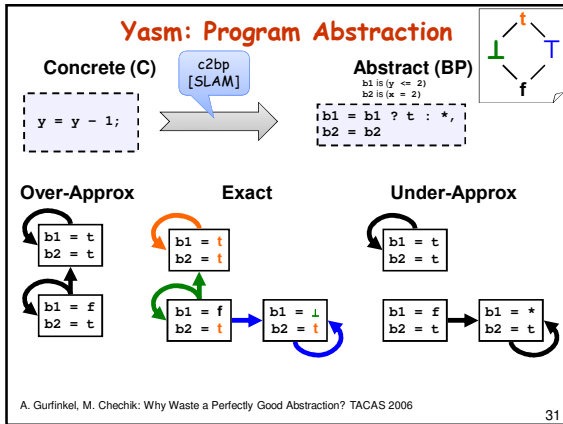
29

Yasm: Models and Model Checking

- ↳ Belnap Logic
 - ↳ 4-valued logic: true, false, unknown, inconsistent
- ↳ Belnap Kripke Structures
 - ↳ Kripke structures extended to Belnap Logic
 - ↳ Propositions
 - ↳ True, False, or Unknown
 - ↳ Transitions
 - ↳ necessary: T, possible: ⊥
 - ↳ necessary and possible: t, impossible: f
- ↳ Analysis via Multi-Valued Model Checking

30

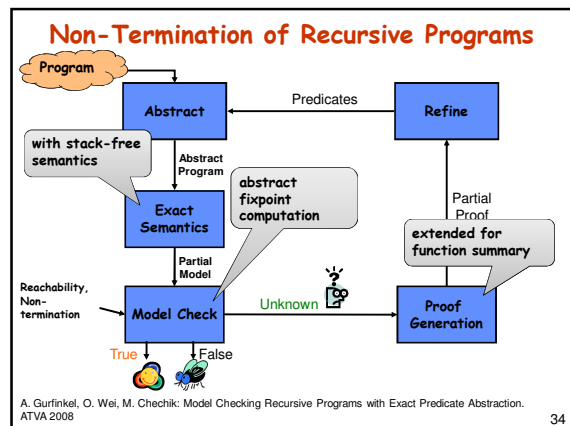
M. Chechik, B. Devereux, S. Easterbrook, A. Gurfinkel: Multi-valued symbolic model-checking. ACM Trans. Softw. Eng. Methodol. 12(4)



Exploiting Exact Approximation Semantics

- ⇒ **Aggressive Abstractions**
 - ↳ Enable coarse and computationally cheaper abstraction
- ⇒ **Shallow Counterexamples**
 - ↳ Refinement guided to promising counterexample
- ⇒ **Reusing Previous Model-Checking Results**
 - ↳ $EF \text{ ERROR} \rightarrow EF(\text{ERROR})$ "unavoidable" states

A. Gurfinkel, O. Wei, M. Chechik: Yasm: A Software Model-Checker for Verification and Refutation. CAV 2006
A. Gurfinkel, M. Chechik: Why Waste a Perfectly Good Abstraction? TACAS 2006 33



Experiments: Non-termination

```
void quicksort(
1: int left, int right){
2: if(left>=right)
3: return;
4: lo=left; hi=right;
5: while(lo<=hi){
6: if(nondet()){
7: lo++;
8: } else {
9: if(lo!=left || hi!=right)
10: hi--;
11: }
12: }
13: quicksort(left, hi);
14: quicksort(lo, right);
15: }
```

Buggy Quicksort

- example from Moped [ES01] and Vera [ACEM05]
 - can prove non-termination over **finite** data domain
 - assume over-approximation for infinite domain
- ☀ **YASM** automatically proves non-termination on **infinite** data domain!
- predicates

```
hi=right, lo=left, lo<=hi, left>=right
```
- non-terminating path

```
pc=2, left=1, right=2
pc=4, lo=left=1, hi=right=2
.....
pc=2, left=1, right=2
```

35

Outline

- ⇒ **Partial Models for Program Abstraction**
 - ↳ Overview of existing modeling formalisms
 - ↳ Program Abstraction "Problem"
 - ↳ Expressiveness of Partial Models
 - ↳ Model Checking w/ Reduced Inductive Semantics
- ⇒ **Yasm: A Brief Overview**
 - ↳ Models and Model Checking algorithm
 - ↳ Program Abstraction
 - ↳ Abstraction Refinement
 - ↳ Exploiting Partial Model Semantics
 - ↳ Checking Non-Termination of Recursive Programs
- ⇒ **Conclusions and Lessons Learned**

36


Lessons Learned

- Partial models can be used effectively in Software Model Checking. Good fit for CEGAR.
- Using partial models in abstraction is very different from using them for modeling and specification
- 4-valued (and 3-valued) analysis is much easier to explain than the multi-valued foundations behind it!
- Both Abstract Interpretation and Model Checking gave us invaluable insights.
 - It pays off to look at the problem with "both eyes"



37

Status and Future Work

- Yasm - state-of-the-art SMC (5 years ago)
 - still great for trying new ideas and heuristics, BUT
 - tied to an old theorem prover (CVCLite)
 - fairly naive front-end and predicate abstraction algorithm
 - WP-based abstraction-refinement (v.s. interpolant-based)
 - no recursion and reduced semantics in stable release
 - will be happy to help port/use/understand
 - available at <http://www.cs.toronto.edu/~yasm>
- Dynamic analysis with partial models 
 - partial model to capture dynamically explored states
 - under-approximation from dynamic execution
 - over-approximation from abstraction of source code
 - see our paper w/ A. Albarghouthi and O. Wei in CAV 2010

38

Acknowledgements

YASM is a joint work with

- Ou Wei
- Tom Hart (PtYasm)
- Kelvin Ku
- Shiva Nejati

- Laurie Lugin
- Xin (John) Ma



XChek is a joint work with

- Steve Easterbrook
- Benet Devereux
- Mihaela Gheorghiu
- Shiva Nejati
- Albert Lai

- Victor Petrovykh
- Christopher Thompson-Walsh
- Anya Tefliovich

39

References

- Ou Wei, Arie Gurfinkel, Marsha Chechik: *Mixed Transition Systems Revisited*. VMCAI 2009
- Arie Gurfinkel, Ou Wei, Marsha Chechik: *Model Checking Recursive Programs with Exact Predicate Abstraction*. ATVA 2008
- Arie Gurfinkel, Ou Wei, Marsha Chechik: *Yasm: A Software Model-Checker for Verification and Refutation*. CAV 2006
- Arie Gurfinkel, Marsha Chechik: *Why Waste a Perfectly Good Abstraction?* TACAS 2006
- Arie Gurfinkel, Ou Wei, Marsha Chechik: *Systematic Construction of Abstractions for Model-Checking*. VMCAI 2006
- Arie Gurfinkel, Marsha Chechik: *Generating Counterexamples for Multi-valued Model-Checking*. FME 2003
- Arie Gurfinkel, Marsha Chechik: *Proof-Like Counter-Examples*. TACAS 2003
- Marsha Chechik, Benet Devereux, Steve M. Easterbrook, Arie Gurfinkel: *Multi-valued symbolic model-checking*. ACM Trans. Softw. Eng. Methodol. 12(4)

40

Questions?

Comments?

Concerns?

Suggestions?

THANKS FOR YOUR
ATTENTION!