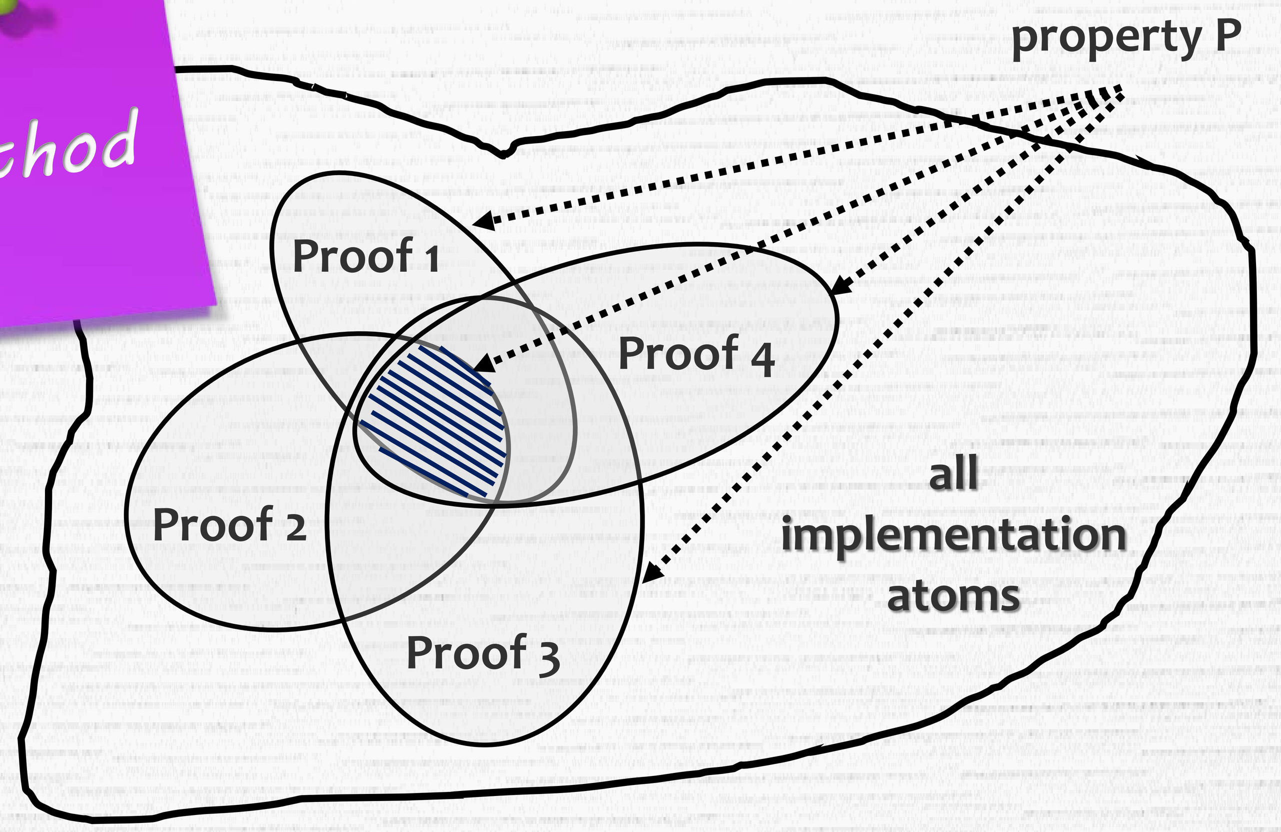
Inductive Validity Cores for Formal Verification

Elaheh Ghassabani, Michael Whalen, and Andrew Gacek

Why is my property valid?

Symbolic model checkers can construct proofs of properties over very complex models. However, the results reported by the tool when a proof succeeds do not generally provide much insight to the user. We introduce *Inductive Validity Cores* (IVCs), minimal sets of model elements necessary to construct *inductive proofs*, such as those constructed by modern model checking algorithms using k-induction and PDR. These IVCs can serve as *explanations* of the proof. We have implemented and evaluated IVCs in the JKind model checker and explored several applications of the idea.

Method



Building Blocks

- Source artifact (requirement or property) $r \in \Delta$
- Target artifacts (implementation or model elements) $S \subseteq \Sigma$
- $S \vdash r$ for $S \subseteq \Sigma$ and $r \in \Delta$ when S satisfies r
  - S is an IVC set
- **MIVC**: a minimal set of target artifacts to construct a proof
  - maps a requirement to an IVC set

$$MIVC(r, s) \equiv s \vdash r \wedge (\neg \exists s_0 .\ s_0 \subset s \wedge s_0 \vdash r)$$

- There could be many IVCs for a property: all IVC sets

$$AIVC(r) \equiv \{ s \mid s \subseteq \Sigma \wedge (r, s) \in MIVC \}$$

Categorizing Target Artifacts

$$MUST(r) = \bigcap AIVC(r)$$
$$MAY(r) = \left(\bigcup AIVC(r)\right) \setminus MUST(r)$$
$$IRR(r) = \Sigma \setminus \left(\bigcup AIVC(r)\right)$$

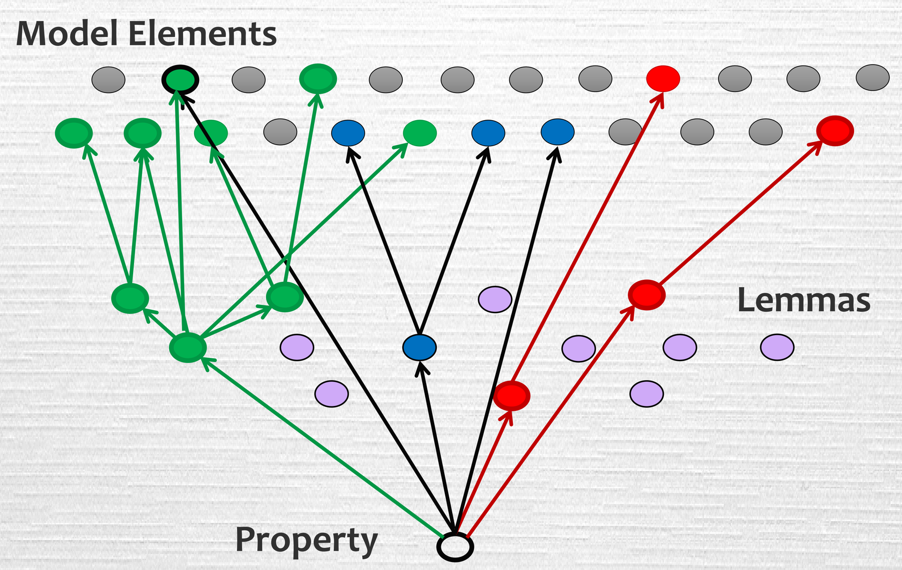
Publications:

1) *Efficient generation of inductive validity cores for safety properties, FSE, 2016*
2) *Complete traceability for requirements in satisfaction arguments, RE@Next, 2016*
3) *A new notion of requirements coverage for formal verification (under submission)*
4) *Efficient generation of all IVCs (in preparation)*

Algorithm

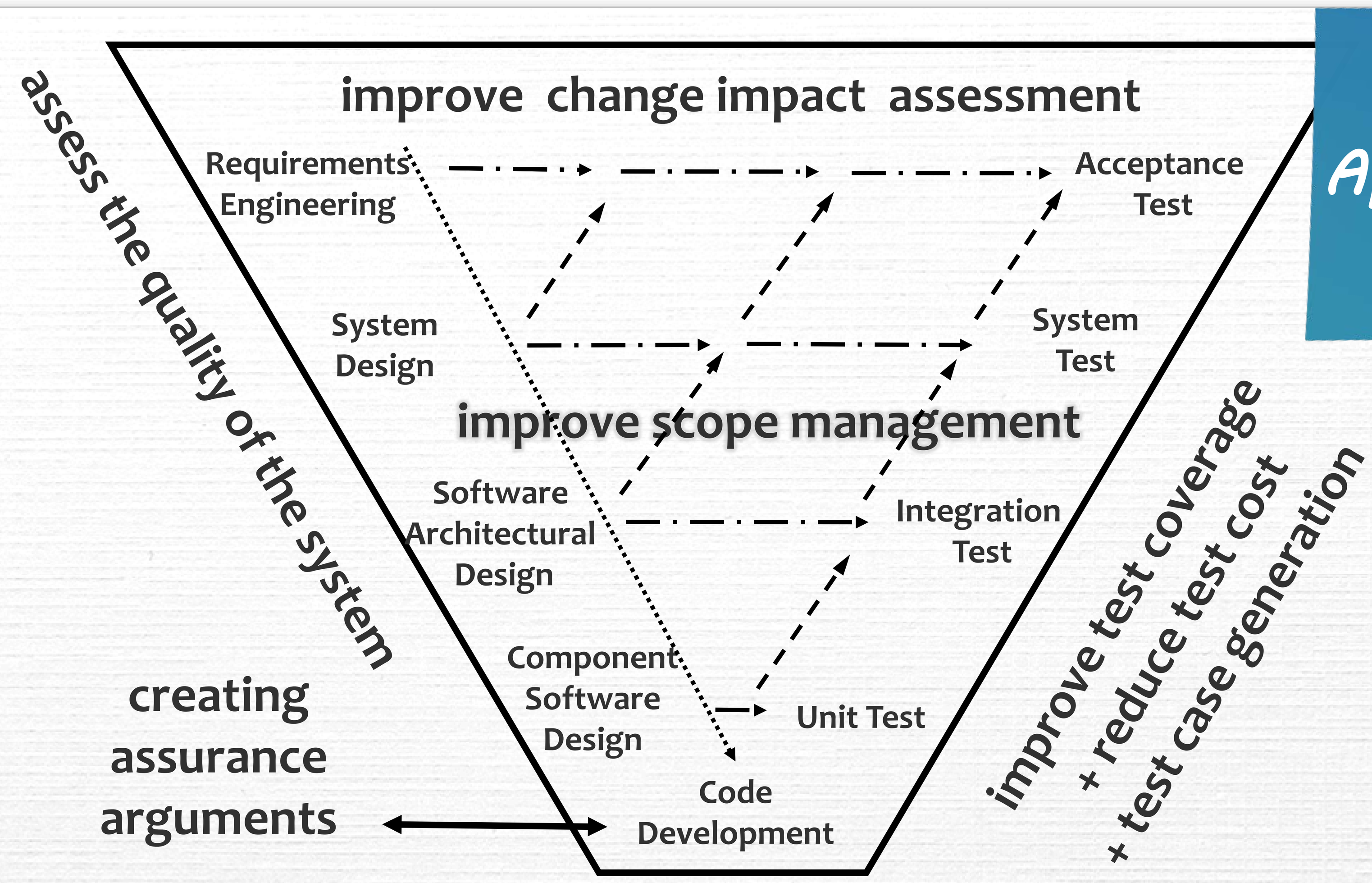
For each valid property:

1) Reduce invariants and find a minimal set of invariants
2) Obtain a minimum K by which the property is K-inductively provable
3) For the obtained K, compute unsat-core with a special IVC query
4) Unsat-core will contain an IVC set
5) Minimize the core to get a minimal IVC set



Applications

- Vacuity Detection
- Checking Requirements Completeness (Coverage Metrics for Formal Verification)
- Requirements Traceability
- Symbolic Simulation/ Test Case Generation
- Explanation of Model Inconsistencies

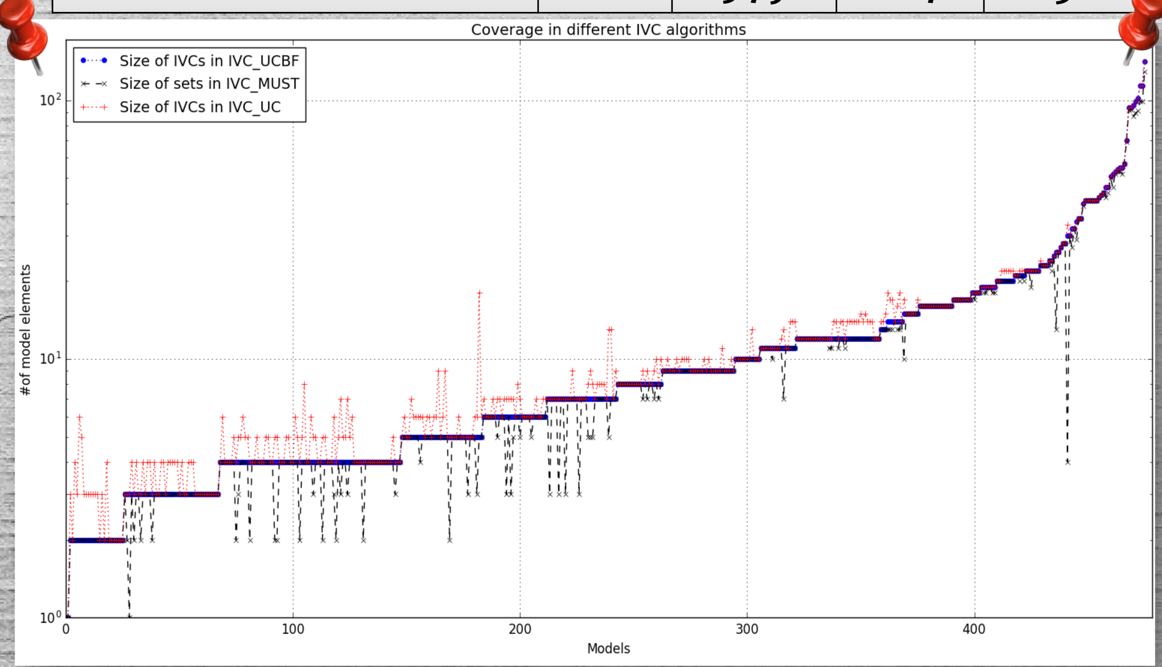


- NASA CVFCS on the Quad-redundant flight controller
- Rockwell Collins, part of the AFRL SpEAR project
- Helicopter architecture proofs in the DARPA HACMS project
- NSF Medical device project on the GPCA model
- AGREE Symbolic Simulator, Part of DARPA SOSITE project
- AGREE/JKind Test-Case Generator

Experiments:
476 LUSTRE models
Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz 3.50 GHz + 16 GB RAM
JKind experimental branch

Running time

Running time (sec)	min	max	avg	stdev
All_IVCs + proof time	0.25	2388.50	60.18	257.73
minimization + All_IVCs + proof	0.25	2388.50	60.26	257.86
IVC_UC + proof time	0.062	14.758	1.38	2.04
IVC_UCBF + proof time	0.248	1323.51	17.24	104.83
IVC_MUST + proof time	0.25	1010.82	20.64	98.97
Proof time	0.047	14.617	1.299	1.94
All_IVCs	0.125	2375.0	58.88	256.52
IVC_UC	0.0	1.42	0.08	0.18
minimization	0.0	5.79	0.07	0.36



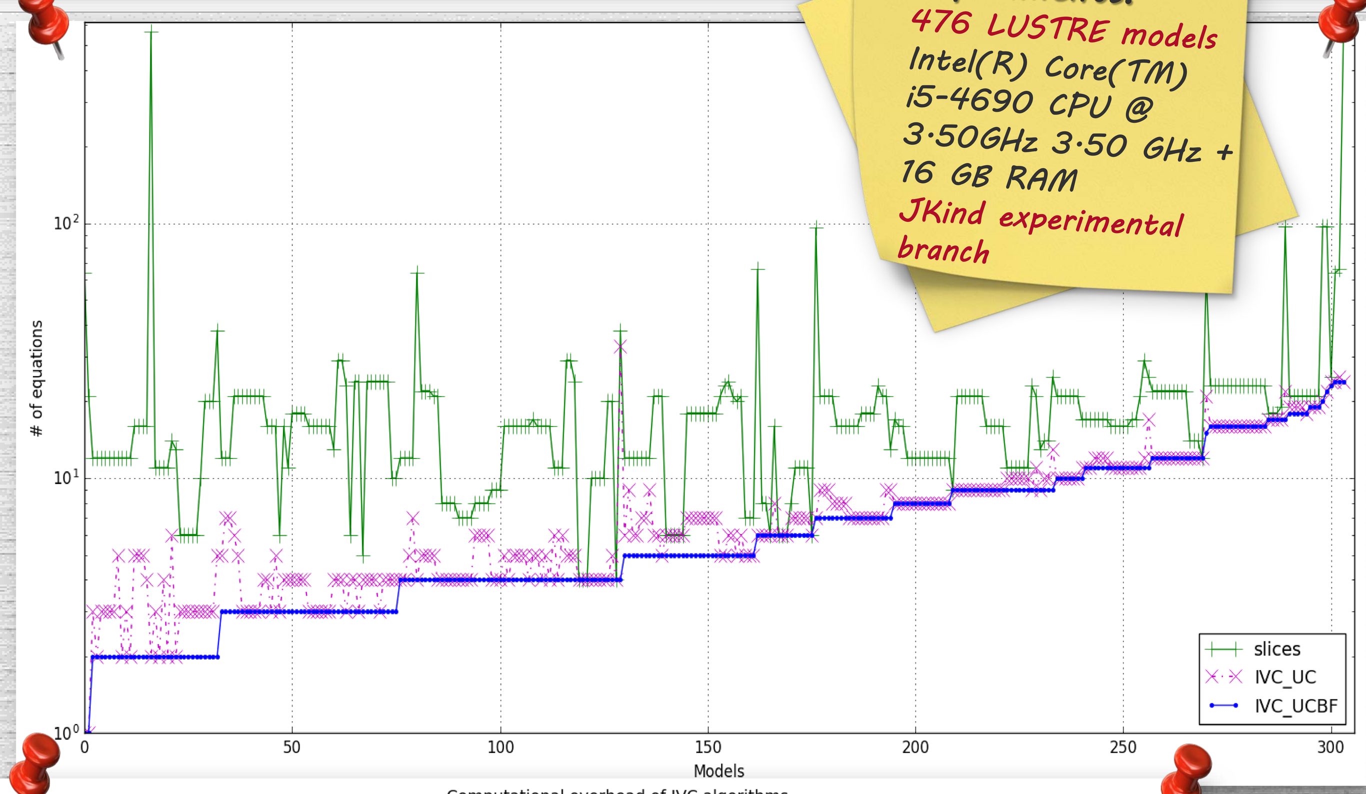
Overhead

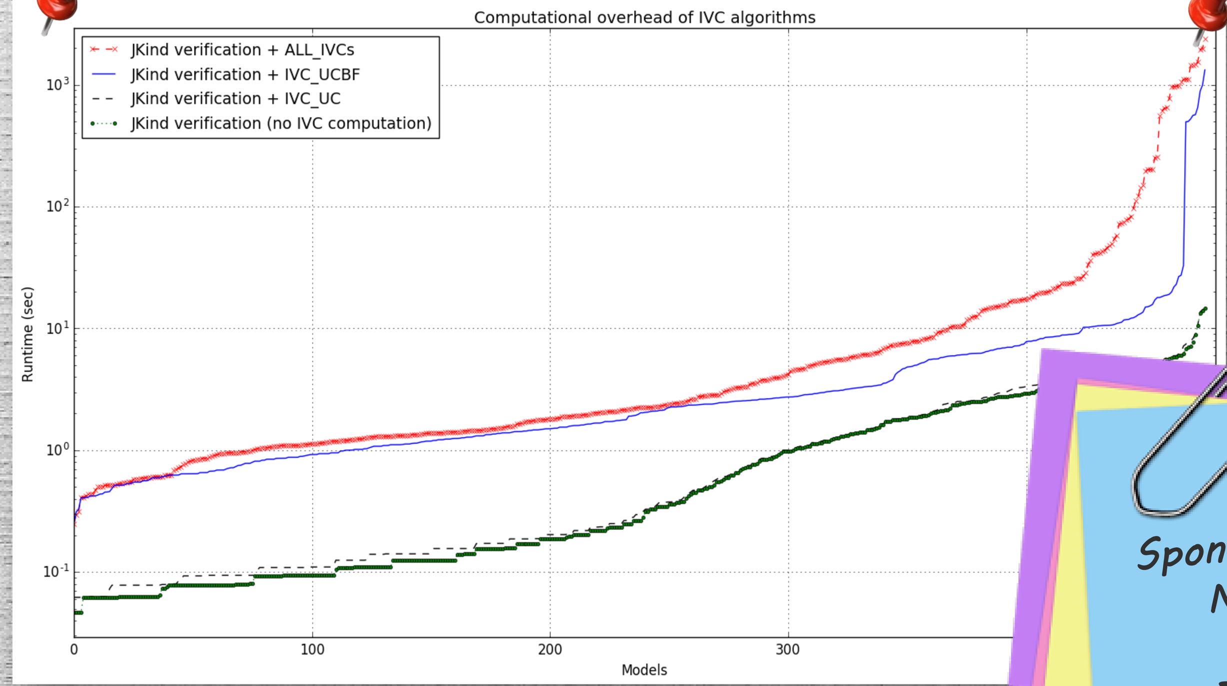
Algorithm	min	max	avg	stdev
All_IVCs	13.6%	101034%	2544%	7764%
Minimization	0.0%	3646%	26%	181%
IVC_UC	0.0%	100%	10%	11%
IVC_UCBF	14.1%	11124%	882%	1512%
IVC_MUST	13.7%	10530%	1081%	1613%

Coverage of different algorithms

IVC sizes	min	max	avg	stdev
IVC_UC	1	141	12.741	15.986
IVC_UCBF	1	141	12.174	16.092
IVC_MUST	1	129	11.644	15.027

Ratio of IVC sizes	min	max	avg	stdev
IVC_UC to IVC_UCBF	100%	360%	110.5%	23.0%
IVC_MUST to IVC_UCBF	13.3%	100%	95.9%	12.1%
IVC_UC to IVC_MUST	100%	825%	119.5%	47.3%





Sponsored by NASA & DARPA