Lazy Abstraction and SAT-based Reachability in Hardware Model Checking

Yakir Vizel

Orna Grumberg

Sharon Shoham

FMCAD 2012

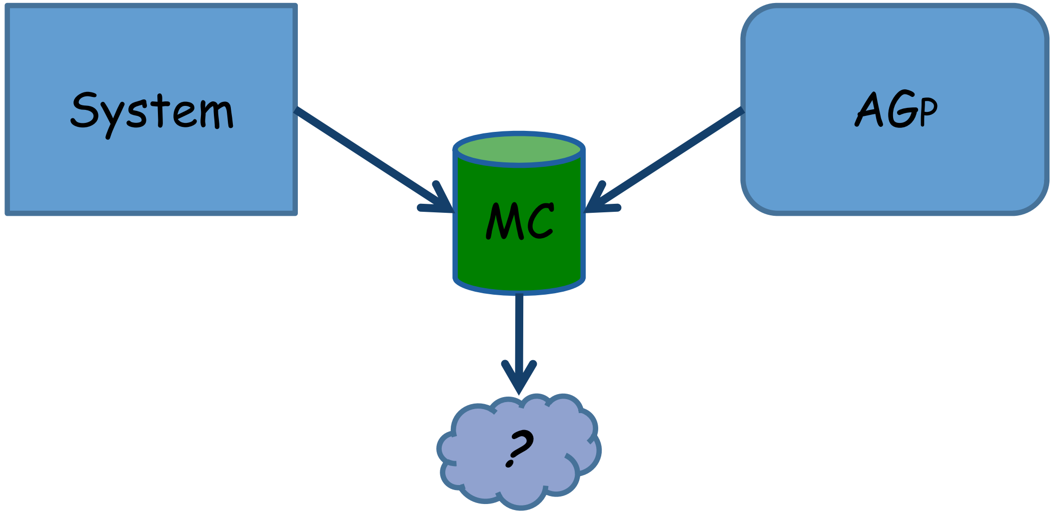
Computer Science Department, Technion, Israel

Outline

- Background
  - Reachability Analysis
  - Abstraction
  - Lazy Abstraction
  - IC3
- Lazy Abstraction with IC3

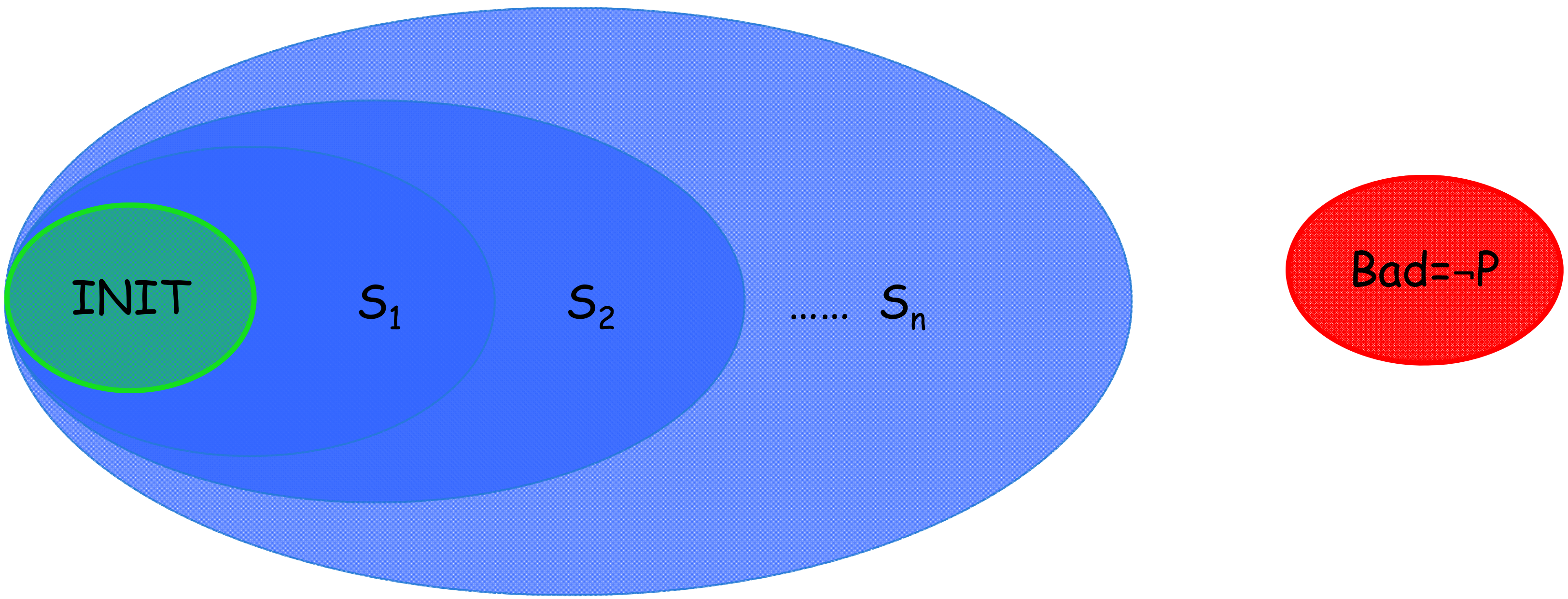
Model Checking

- Given a system and a specification, does the system satisfy the specification.



Reachability Analysis

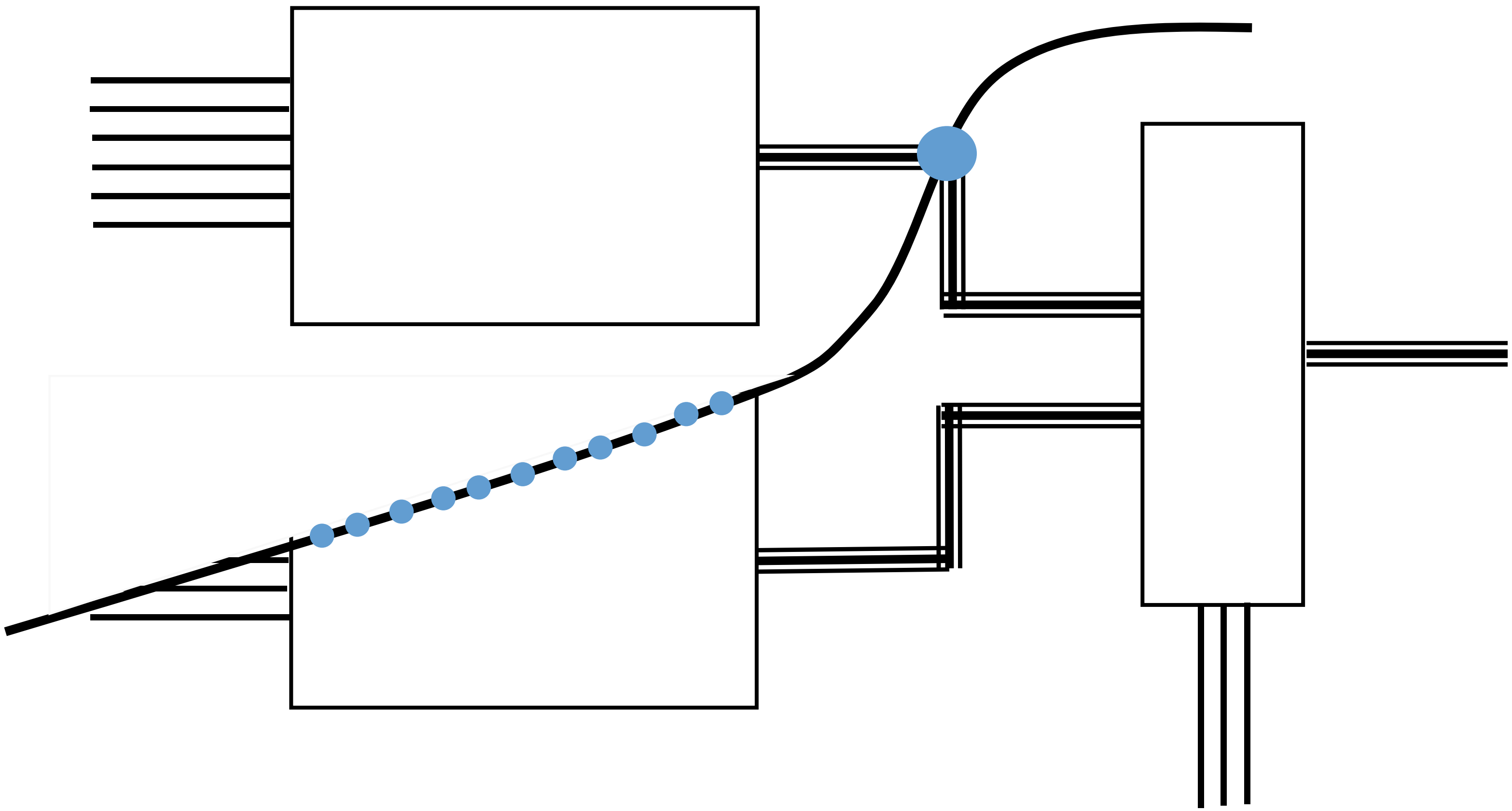
Does AGp hold?



Abstraction

- Fights the state explosion problem
- Removes or simplifies details that are irrelevant
- Abstract model contains less states

Visible Variables Abstraction



Abstraction-Refinement

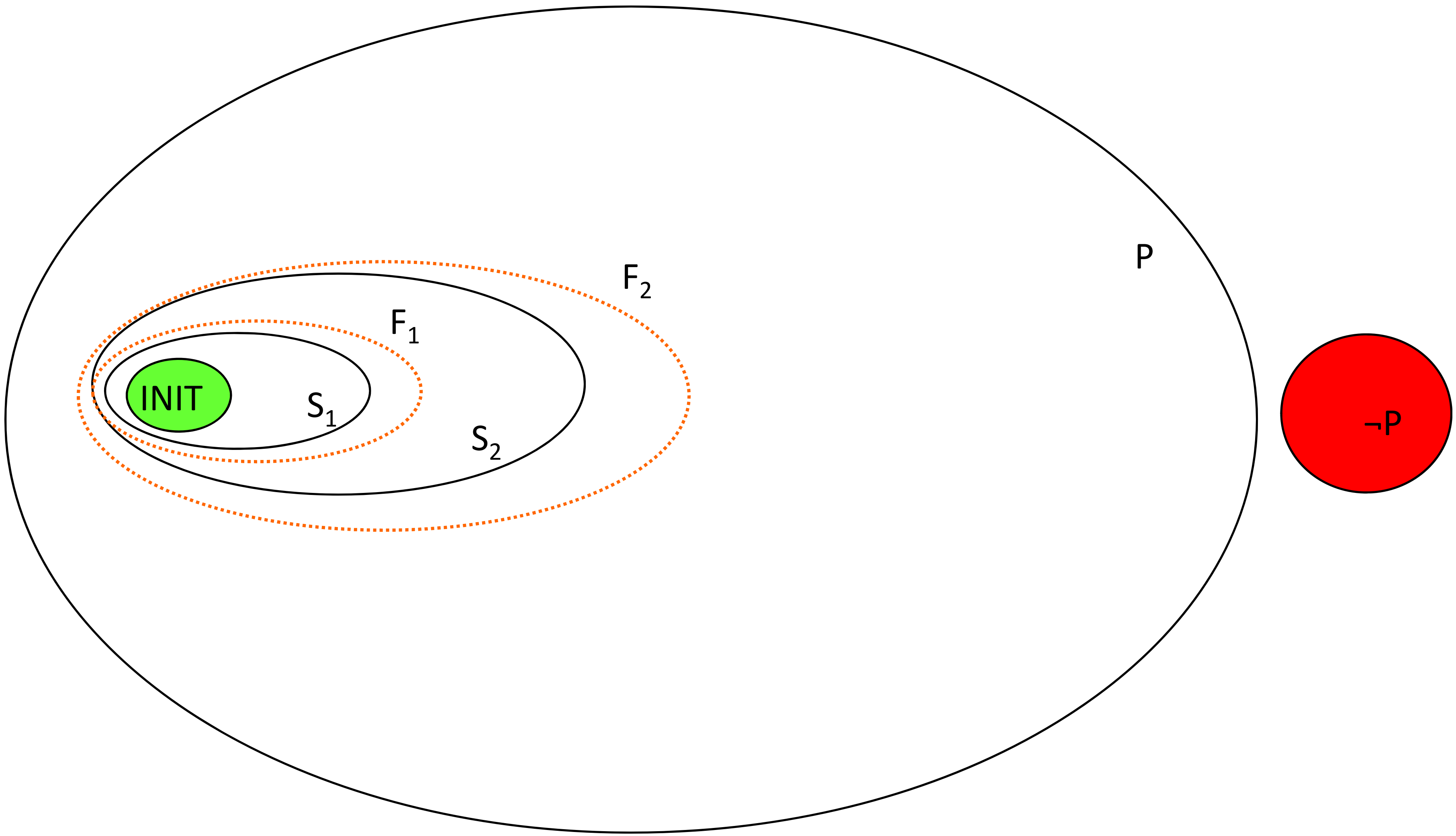
- Abstract model may contain spurious behaviors
  - Spurious counterexample may exist
- Refinement is applied to remove the spurious behavior

Lazy Abstraction

- Different abstractions at different steps of verification

- Refinement applied locally, where needed

SAT-based Reachability with IC3

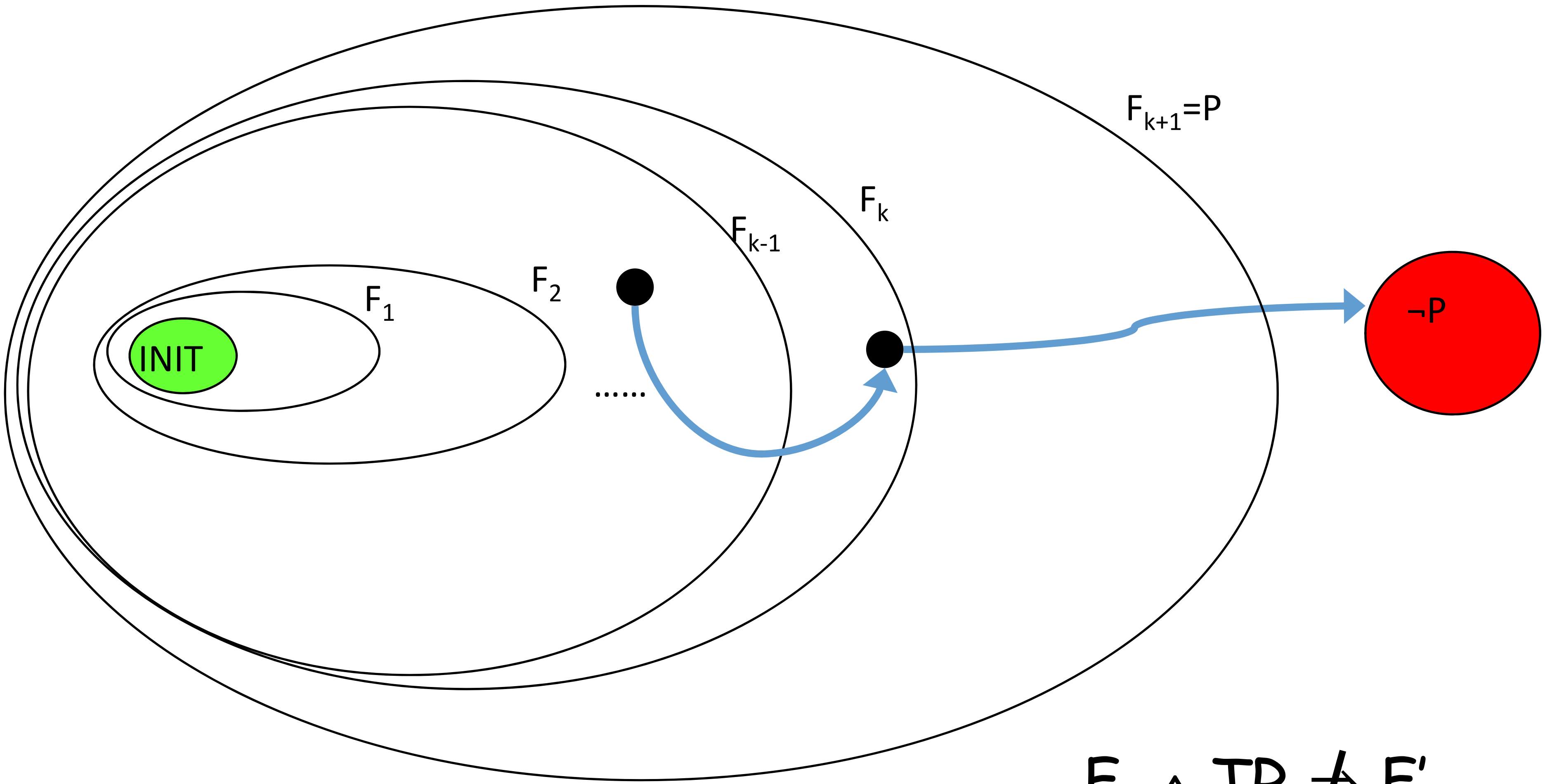


IC3 Basics

- Iteratively compute Over-approximated Reachability Sequence (OARS) $\langle F_0, F_1, \ldots, F_k \rangle$ s.t.
  - $F_0 = INIT$
  - $F_i \Rightarrow P$
  - $F_i \Rightarrow F_{i+1}$
  - $F_i \wedge TR \Rightarrow F'_{i+1}$

- F_i - CNF formula represented by a set of clauses
- TR – the concrete transition relation
- F'_{i+1} is over the next state variables

Iteration of IC3



$$F_k \wedge TR \not\Rightarrow F'_{k+1}$$

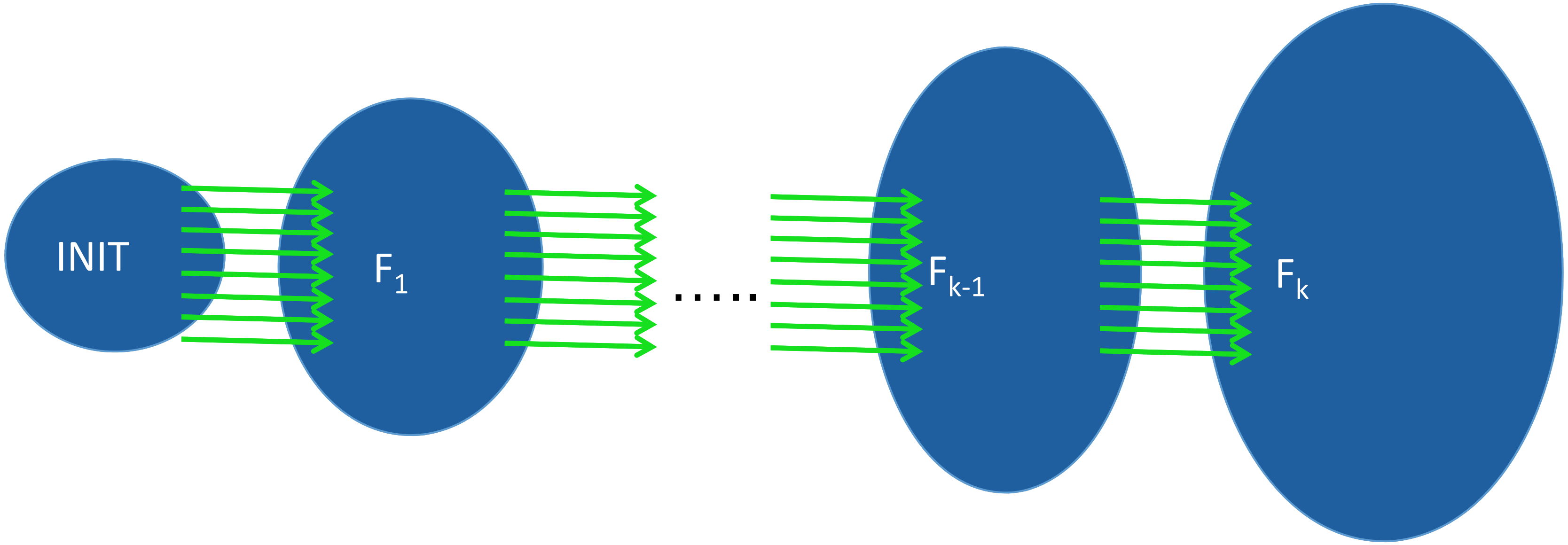
Locality in IC3

- IC3 applies checks of the form
  - $F_k \wedge TR \wedge \neg P'$
    - Finds a state in F_k that can reach $\neg P$
  - $F_i \wedge TR \wedge s'$
    - Finds a predecessor in F_i to the state s
- Using only one TR
  - No unrolling

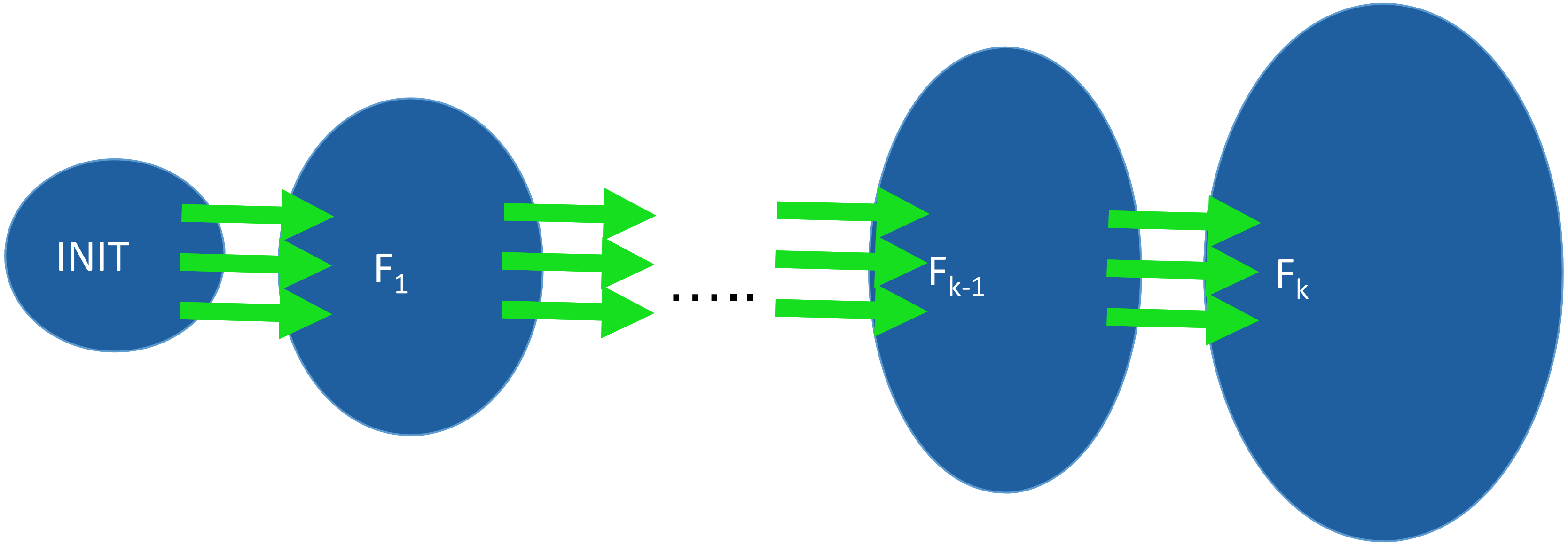
Our Approach - L-IC3

- Use IC3's local checks for *Lazy Abstraction*
  - Different abstraction at different time frames
  - Use visible variables abstraction
  - Different variables are visible at different time frames

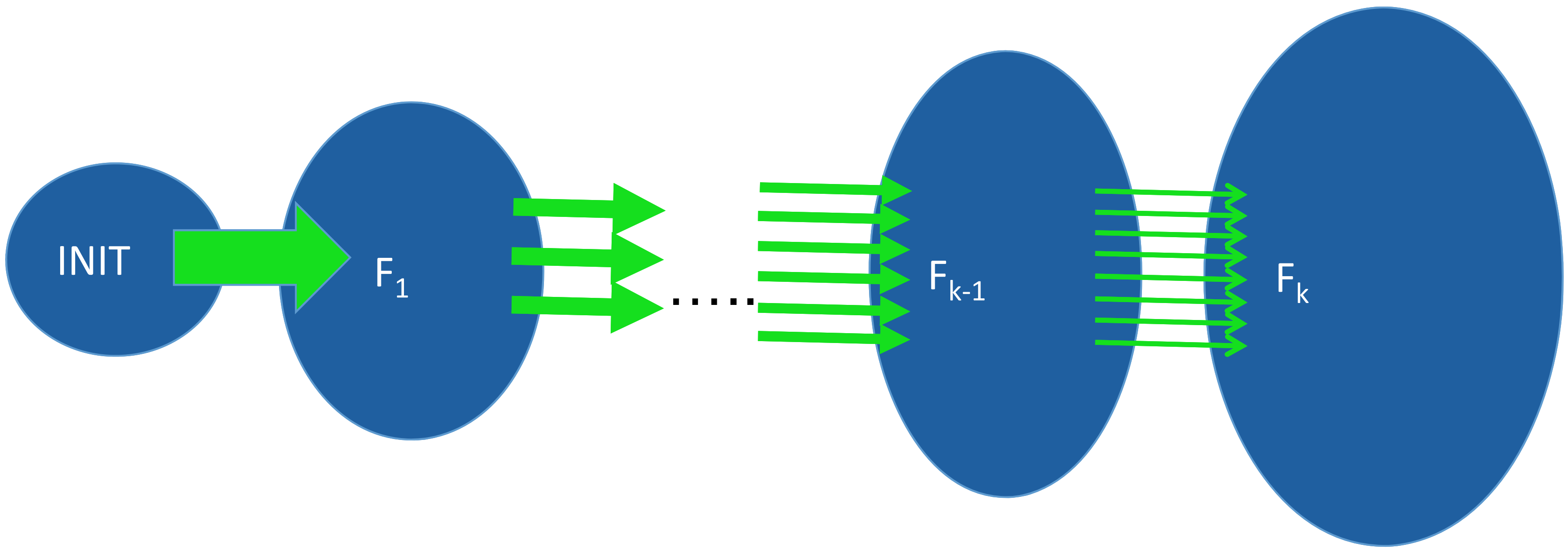
Concrete Model



Using Abstraction



Using Lazy Abstraction



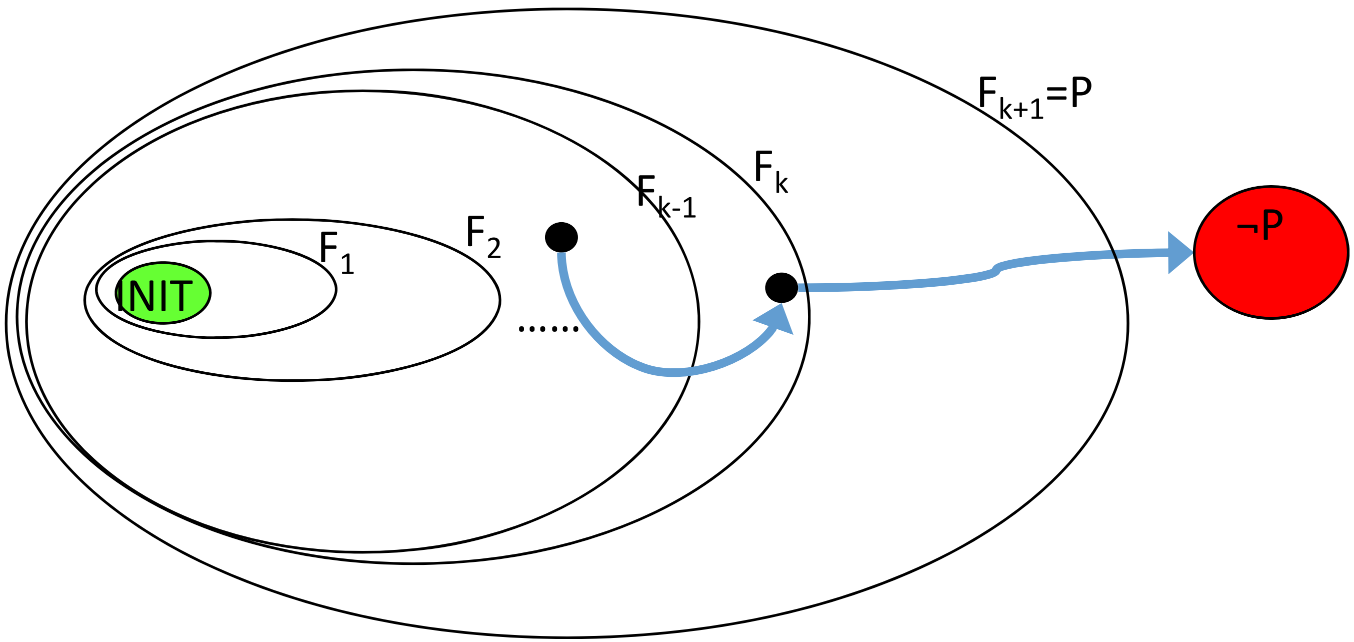
Lazy Abstraction + IC3 = L-IC3

- $\langle F_0, F_1, \dots, F_{k+1} \rangle$ - Reachable states

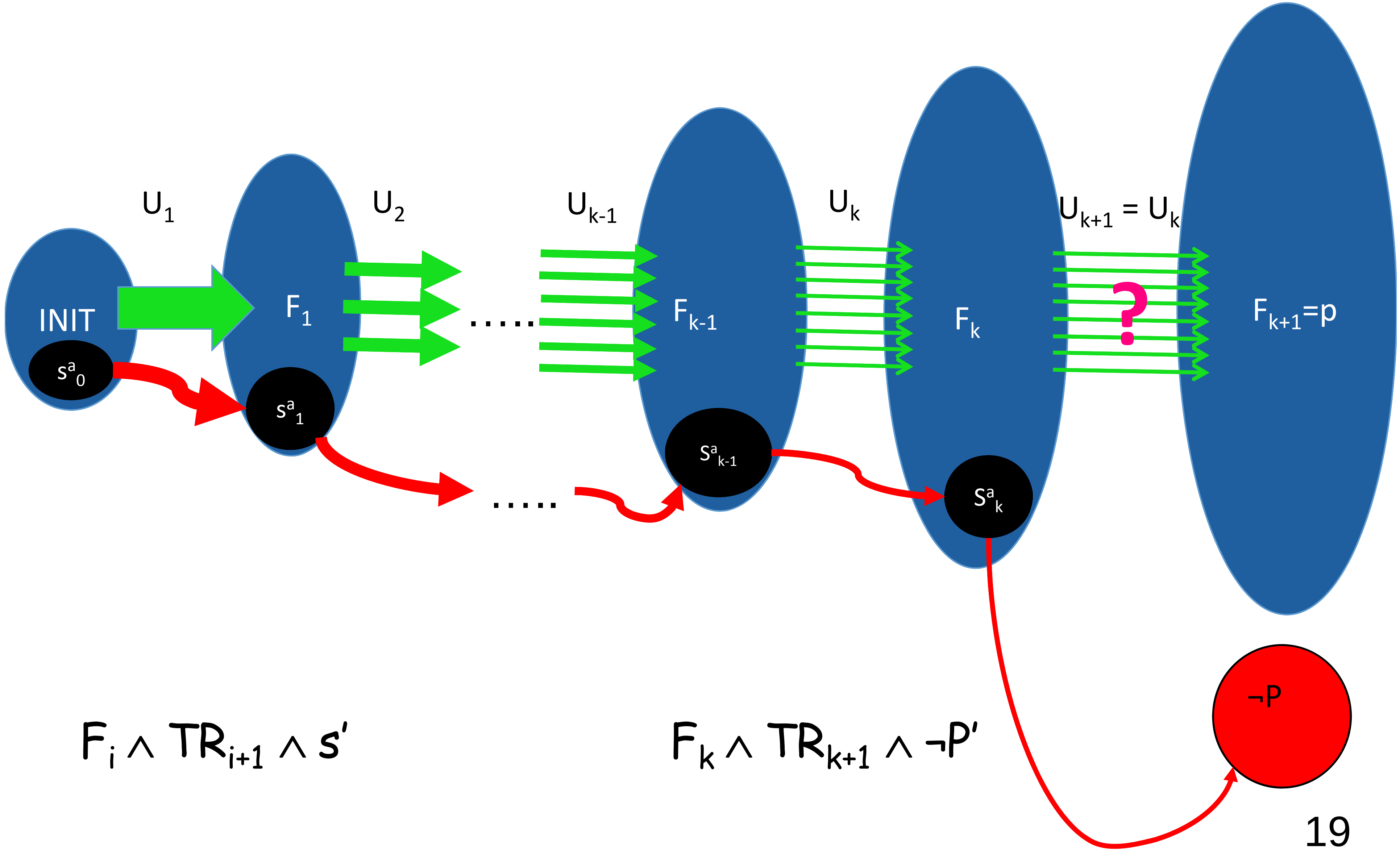
- $\langle U_1, U_2, \dots, U_{k+1} \rangle$ - Abstractions
  - U_i - set of visible variables
    - U_i variables have a next state function
    - The rest, inputs
  - $U_i \subseteq U_{i+1}$
    - U_{i+1} is a refinement of U_i

L-IC3 Iteration

- Initialize F_{k+1} to P
- Initialize U_{k+1} to U_k
- Same problem, the sequence may not be an OARS



Abstract Counterexample

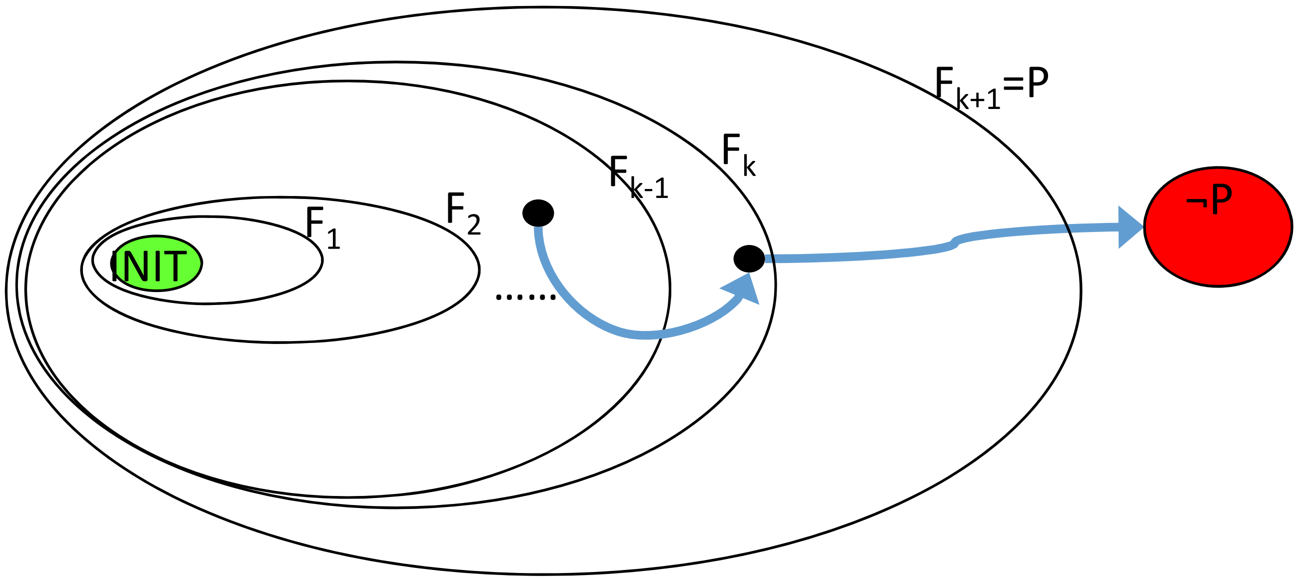


$F_i \wedge TR_{i+1} \wedge s'$

$F_k \wedge TR_{k+1} \wedge \neg P'$

Check Spuriousness

- An abstract CEX of length k+1 exists
- Use an IC3 iteration with the concrete TR
- If a real CEX exists, it will be found



Check Spuriousness (2)

- If no real CEX exists:
  - Compute a *strengthened* sequence $\langle F^r_0, F^r_1, \ldots, F^r_{k+1} \rangle$
  - The strengthened sequence is an OARS
  - Strengthening eliminates all CEXs of length k+1

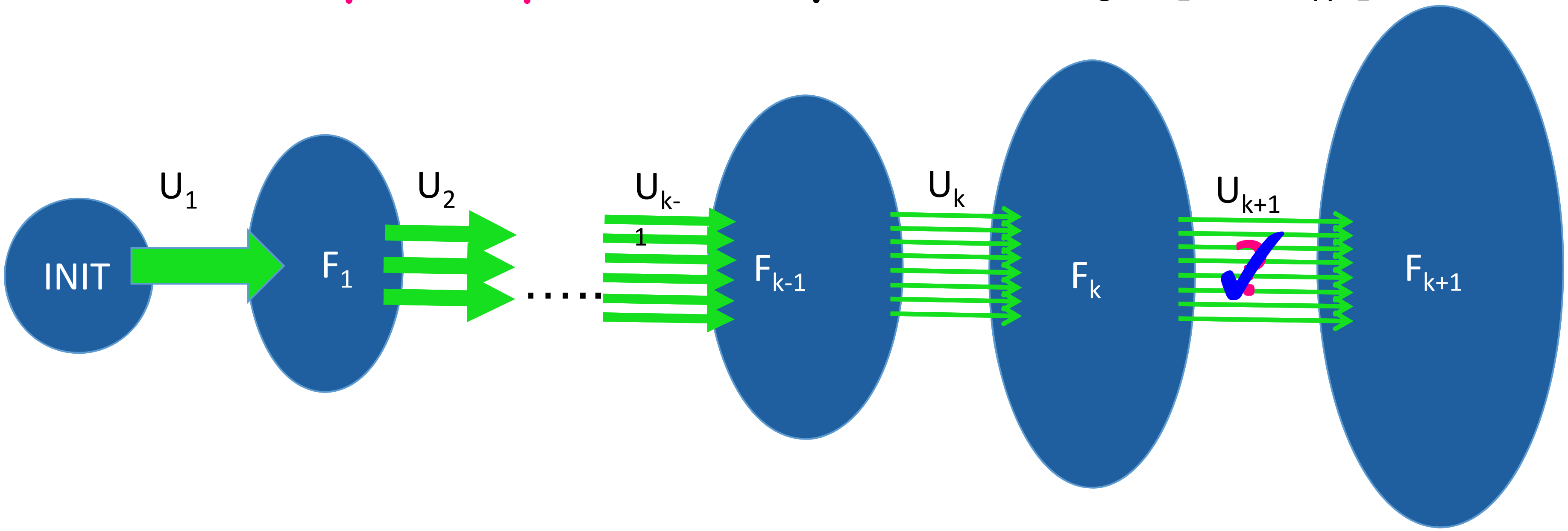
Lazy Abstraction Refinement

- If no real CEX is found by (concrete) IC3 even though (abstract) L-IC3 strengthening failed
  - Abstraction is too coarse

- Refine the sequence $\langle U_1, U_2, \ldots, U_{k+1} \rangle$ as follows:
- Since $F^r_i \wedge TR \Rightarrow F^{r\prime}_{i+1}$
  - $F^r_i \wedge TR \wedge \neg F^{r\prime}_{i+1}$ is unsatisfiable
  - Use the UnSAT Core to add visible variables
    - $U^r_{i+1} = U_{i+1} \cup UCore_i$

Incrementality

- The concrete IC3 iteration works on the already computed sequence $\langle F_0, F_1, \ldots, F_{k+1} \rangle$



- At the end of refinement, L-IC3 continues from iteration k+2

Experiments

Test	#Vars	T/F	#V[Ω]	#V[$Ω_L$]	#C[Ω]	#C[$Ω_L$]	k	k_L	T	T_L
Ind_1	5693	T	236	**11**	617	**62**	14	8	133	**9.2**
Ind_2	*5693*	*T*	*104*	***24***	*2101*	***32***	*32*	*14*	*513*	***13.6***
Ind_3	11866	F	1001	**816**	8457	**3939**	15	18	1646	**599**
Ind_4	*1204*	*T*	*114*	***105***	*18698*	***229***	*8*	*8*	*818*	***3***
Ind_5	3854	T	>470	**666**	>8320	**5363**	>6	11	TO	**730**
Ind_6	*1389*	*F*	***397***	*417*	***12455***	*19742*	*13*	*19*	***262***	*1268*

Experiments - Laziness

Test	#Vars	#TF	#AV	#TF	#AV	#TF	#AV	#TF	#AV	#TF	#AV
Ind_2	5693	1-7	31	8	42	9	51	10-14	54		
Ind_3	11866	1	323	2	647	3	686	4	699	5	705
		6	713	7	714	8	728	9	743		
Ind_5	3854	1	428	2	453	3	495	4	499	5	503
		6	560	7	574	8	576	9-11	577		

Conclusions

- Novel lazy abstraction algorithm for hardware model checking
- Abstraction-Refinement is done incrementally
- More efficient generalization

Up to two orders of magnitude runtime improvement

- Also in the paper: may vs. must proof obligations

Thank You