Predicate Abstraction for Relaxed Memory Models

Andrei Dan
ETH Zurich

Yuri Meshman
Technion

Martin Vechev
ETH Zurich

Eran Yahav
Technion

Motivation

Modern processors' memory operations are not executed in the order specified by the program code

Example:

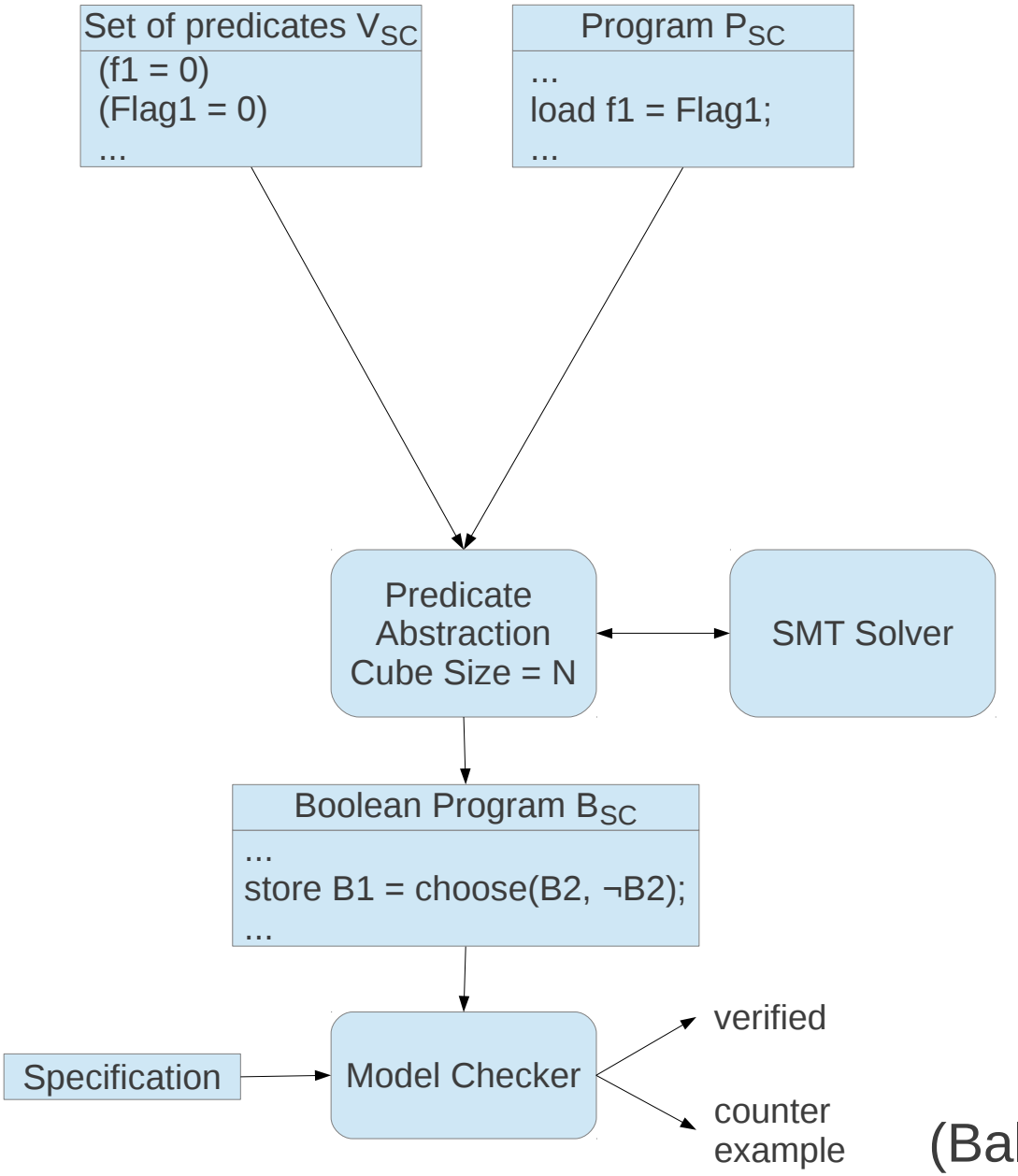
Initial state:
X = 0, Y = 0

Thread 1:	**Thread 2:**
Y = 1;	X = 1;
r1 = X;	r2 = Y;

The final state r1 = 0, r2 = 0 can occur on Intel x86 memory model and cannot occur under SC.

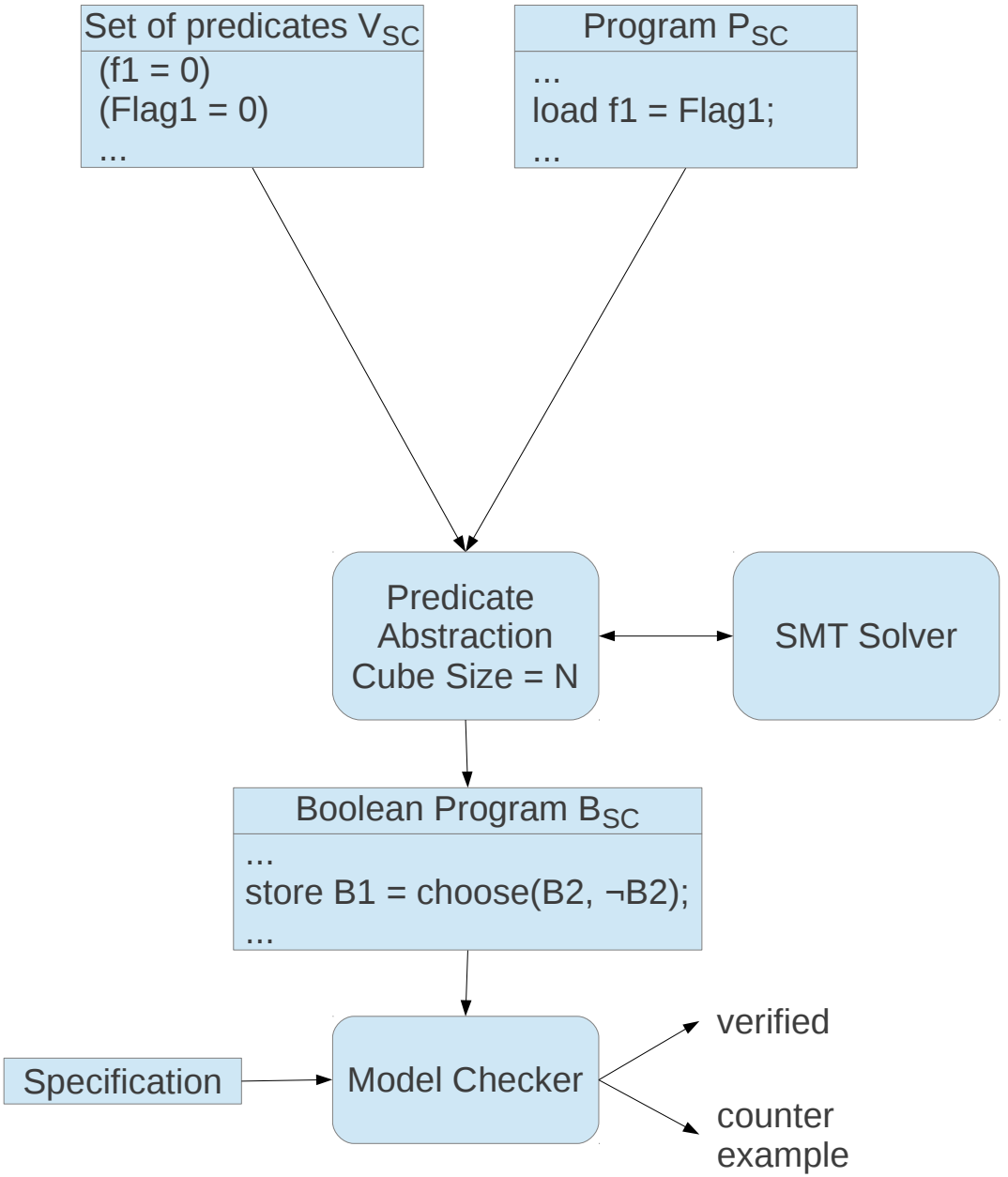
Objective: Automatically verify concurrent programs on relaxed memory models, both finite and infinite state.

Classic predicate abstraction

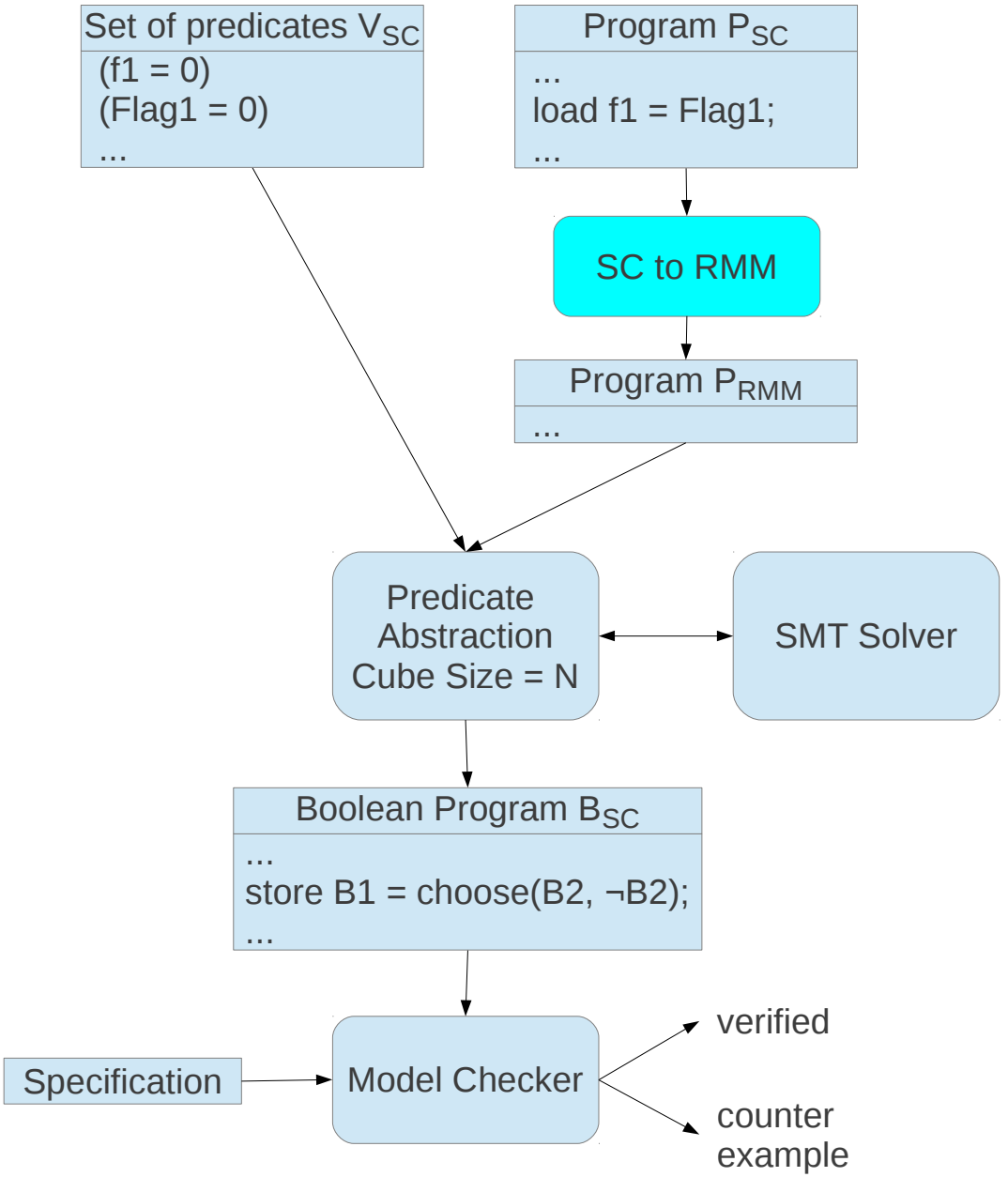


(Ball et al., PLDI '01)

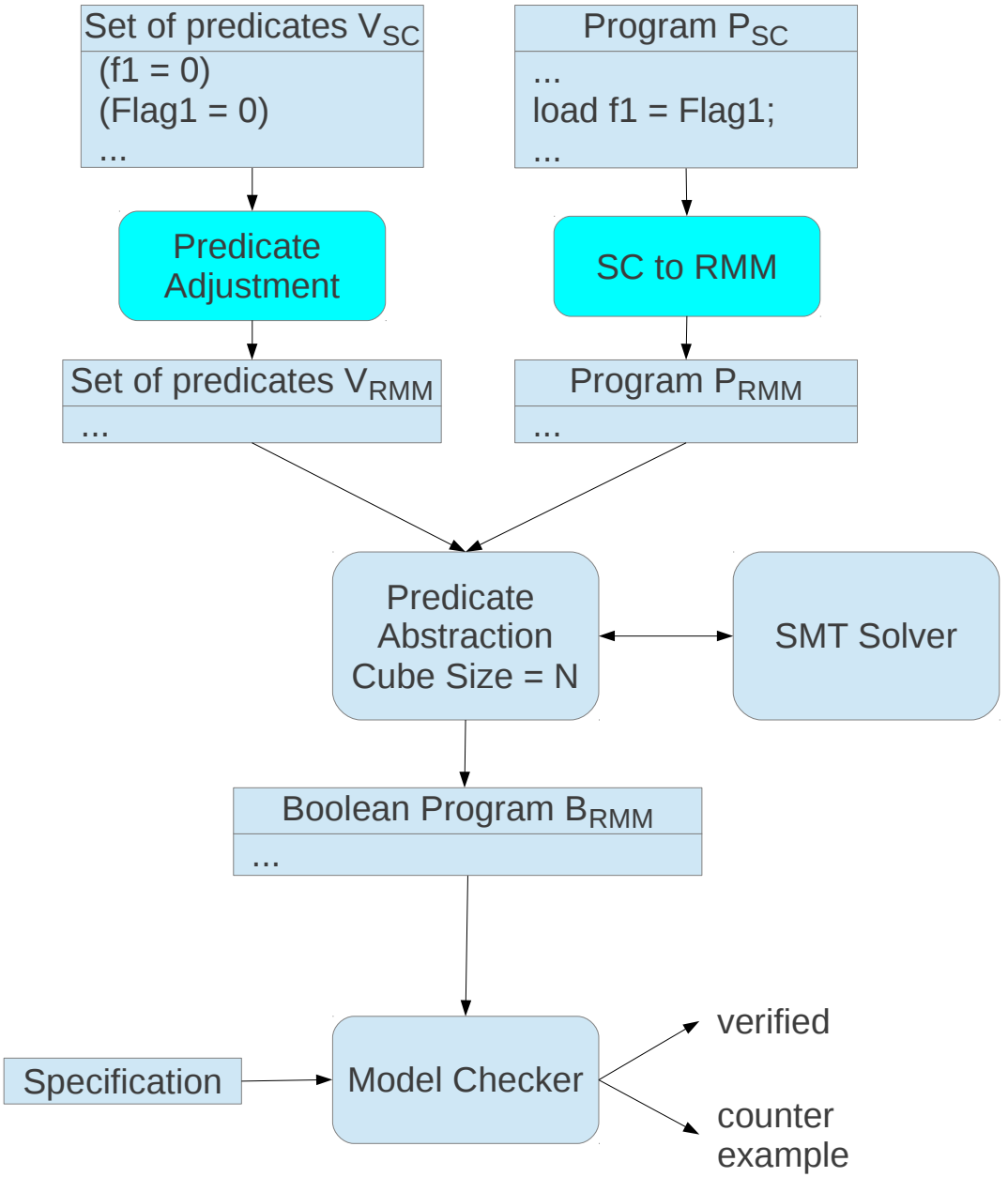
Predicate abstraction for RMM



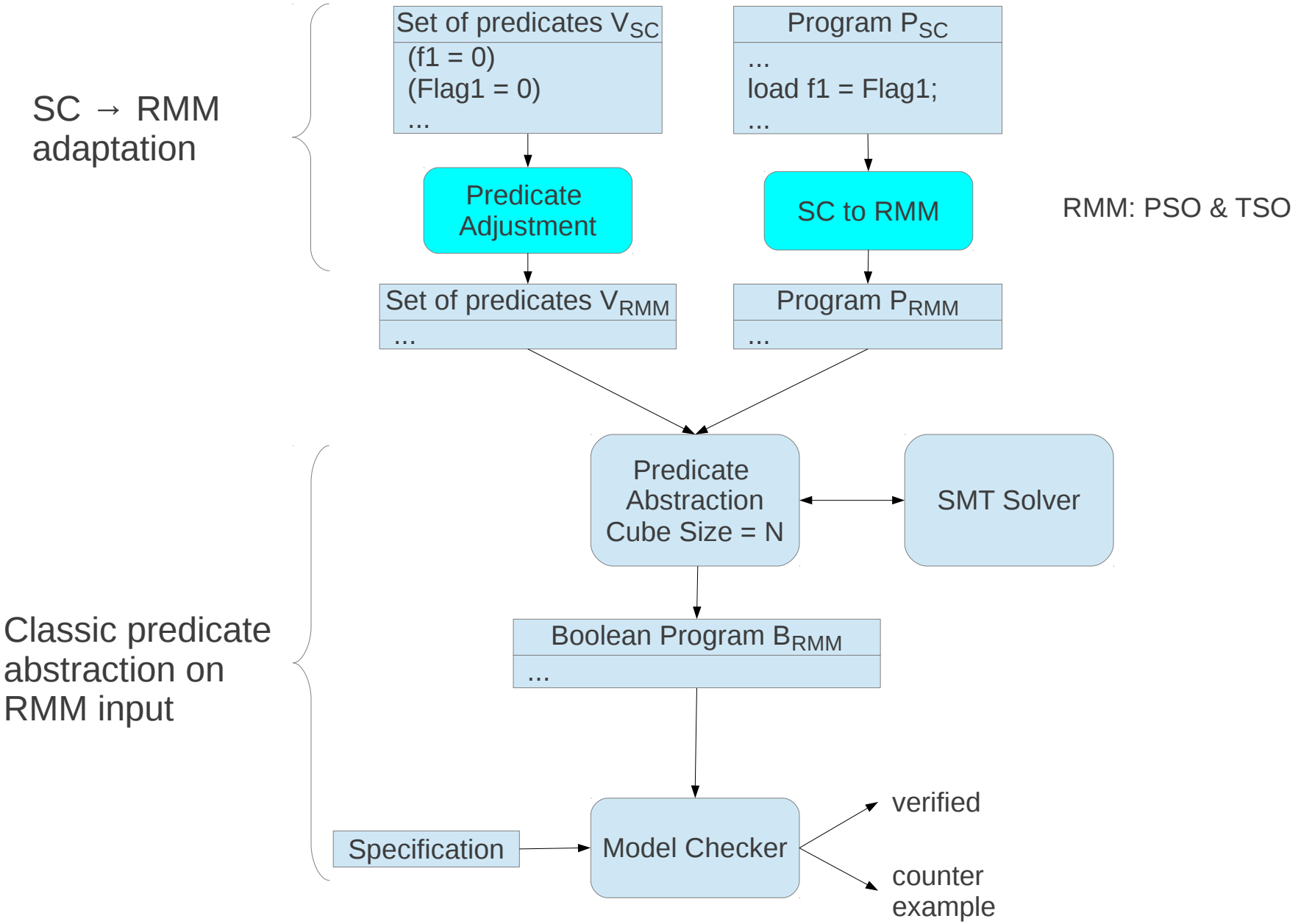
Predicate abstraction for RMM



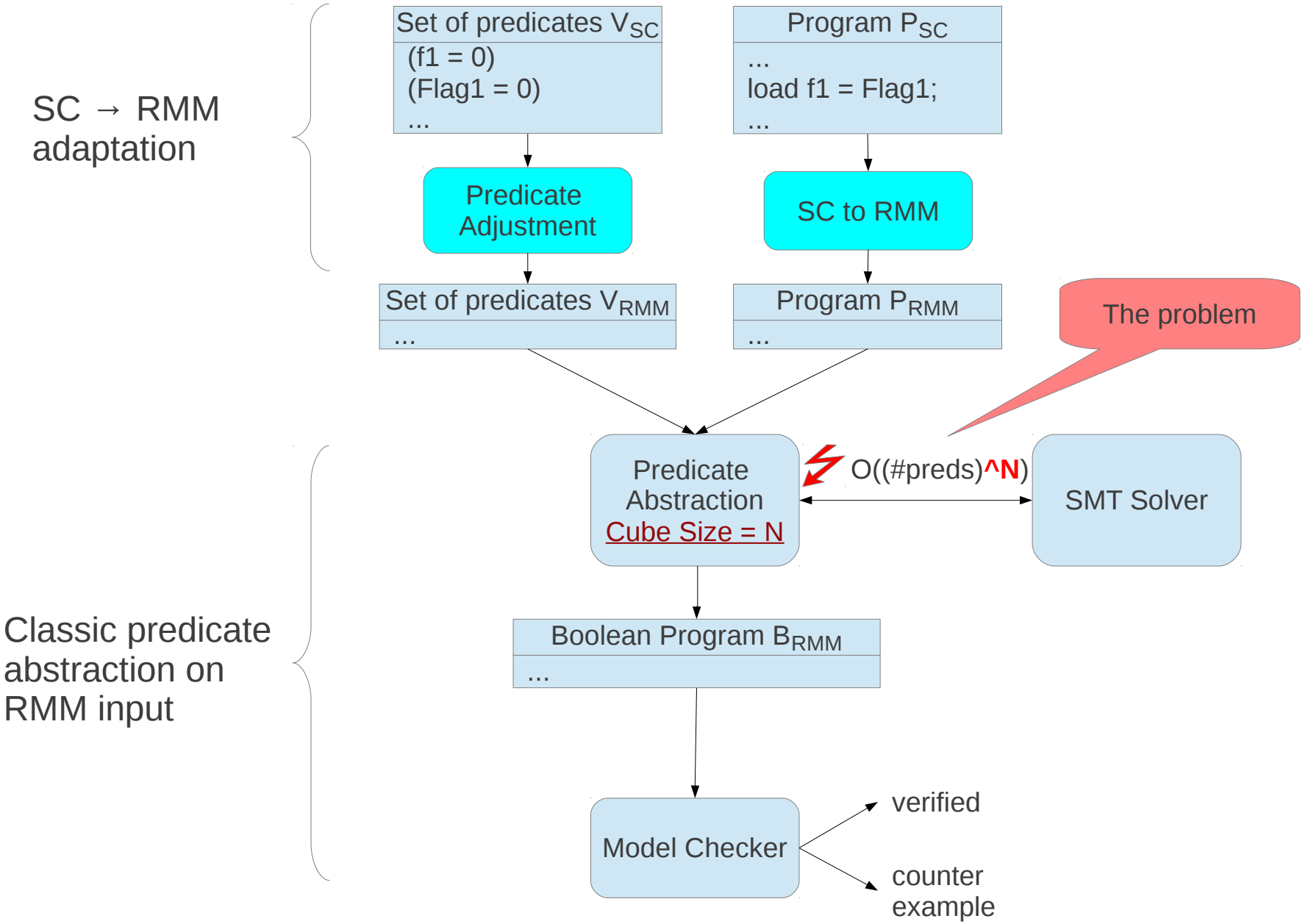
Predicate abstraction for RMM



Predicate abstraction for RMM



Problem: too many calls to the SMT solver



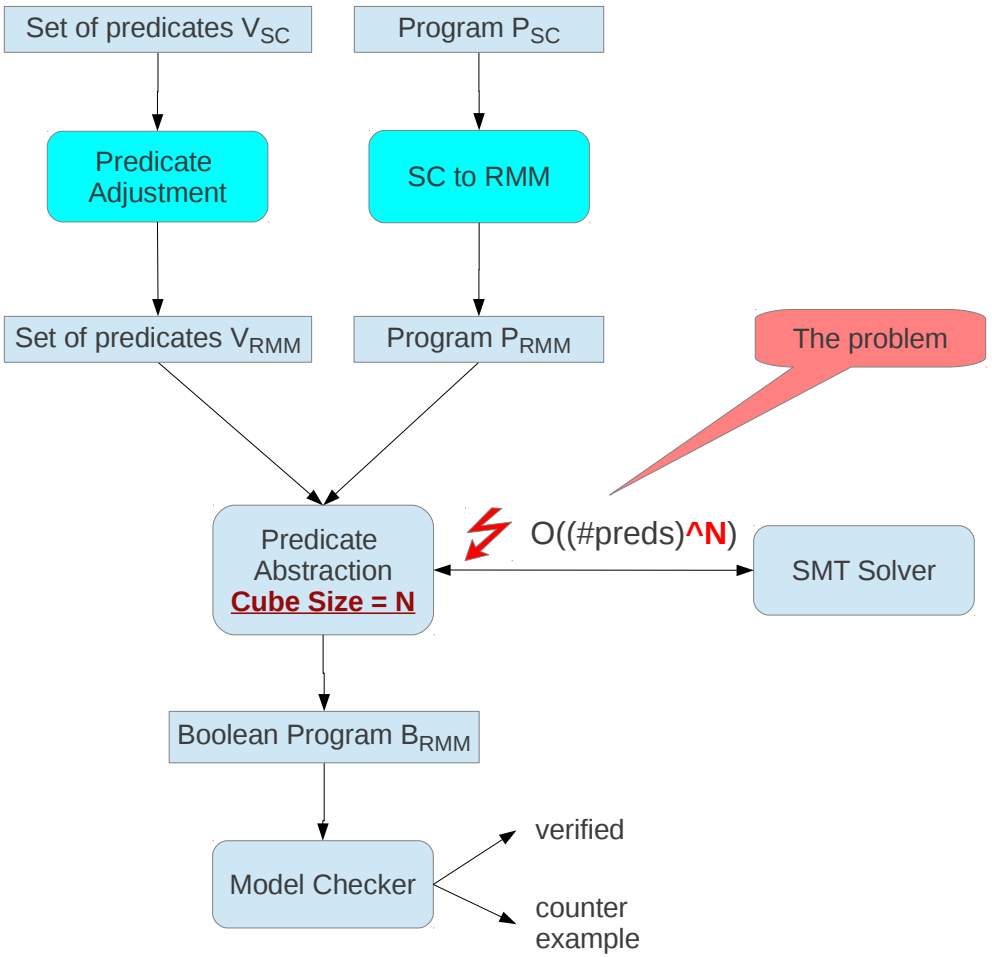
Experimental data for PSO model

Algorithm	Memory model	# predicates	# calls to SMT
ABP	SC	8	4,000
	PSO	15	44,000
Dekker	SC	7	1,500
	PSO	20	102,000
Peterson	SC	7	1,400
	PSO	20	102,000
Bakery	SC	15	1,600,000
	PSO (1 var)	23	91,000,000

For Bakery, the **Cube Size has to be 4** to prove SC correctness.
Building the boolean program for 35 predicates times out.

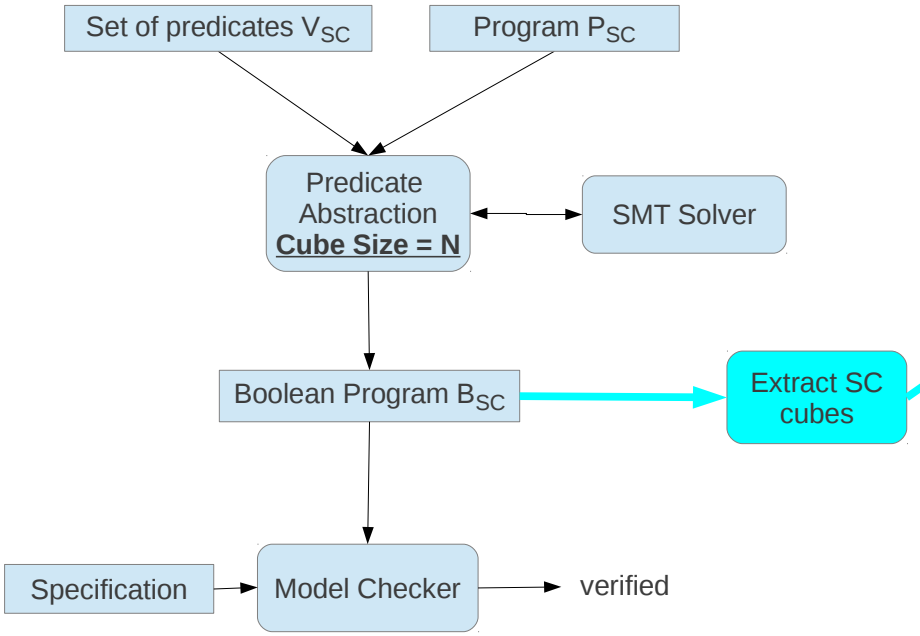
Problem: too many calls to the SMT solver

Build RMM proof:

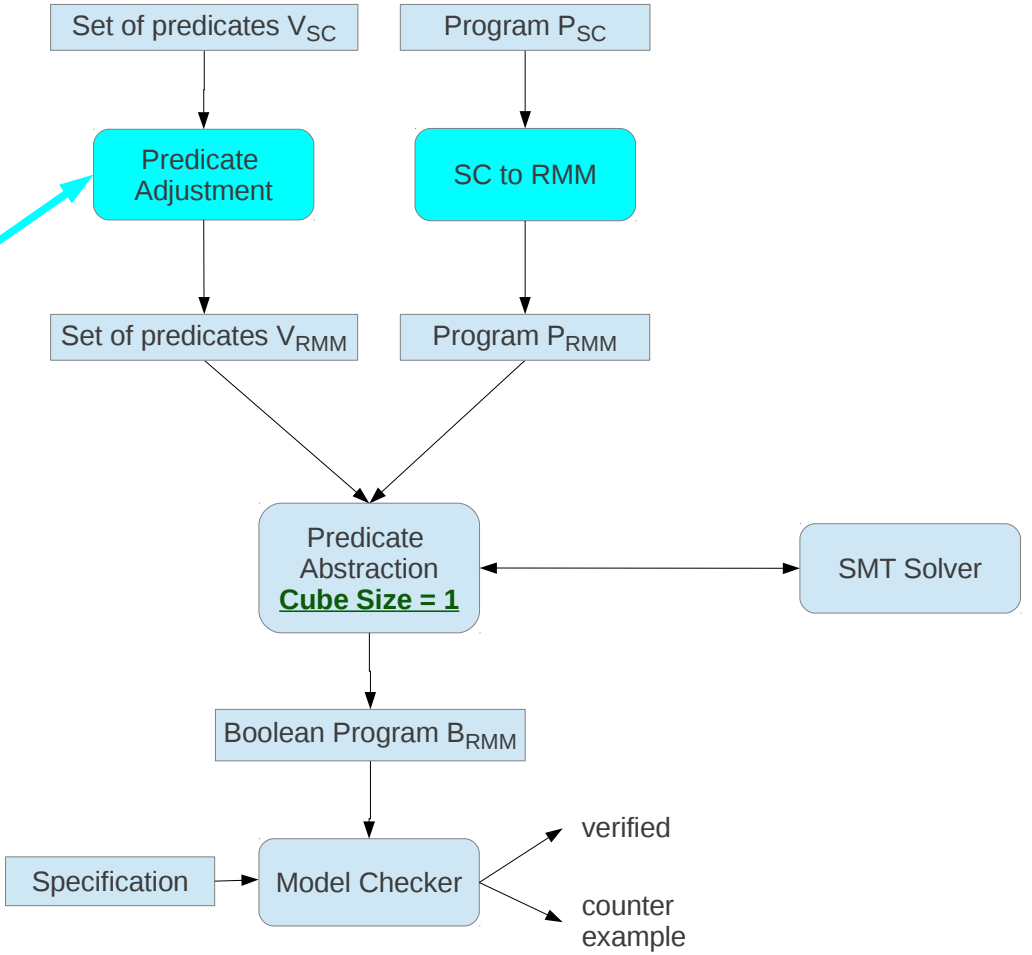


Idea: Leverage the SC proof

Build SC proof:



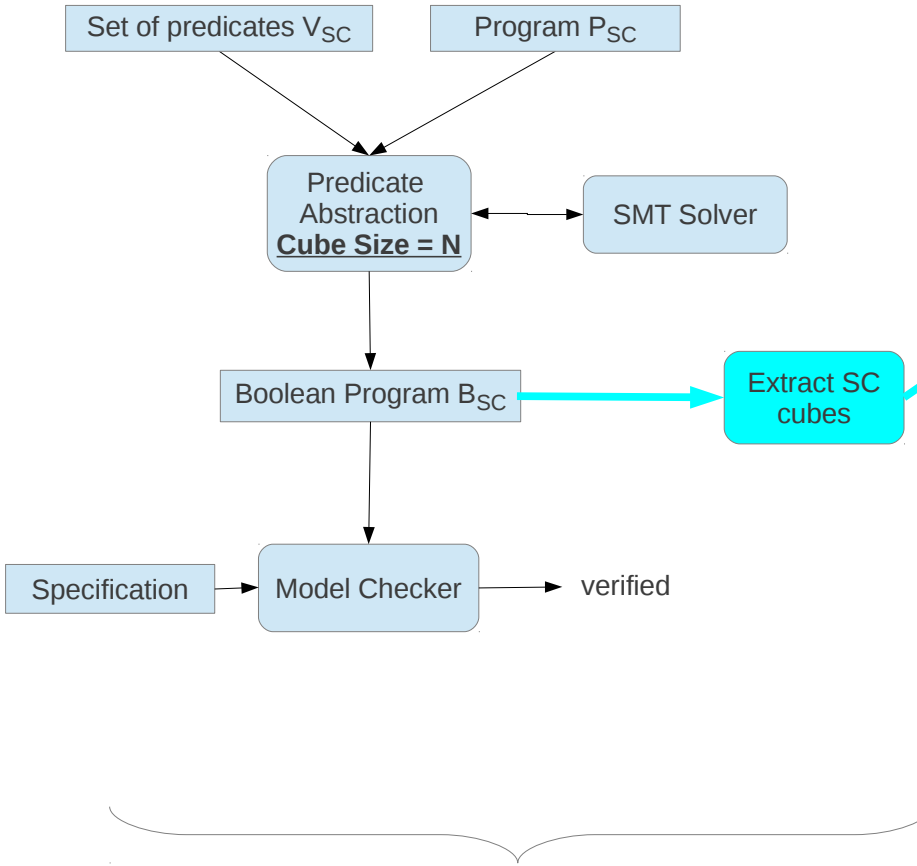
Build RMM proof:



Reuse predicate updating information from SC boolean program

Idea: Leverage the SC proof

Build SC proof:



Build RMM proof:

Reuse predicate updating information from SC boolean program

Results for Bakery 1 variable PSO

	Classic Predicate Abstraction adapted for PSO	Our method: Leverage SC proof	
		Build SC proof	Build PSO proof
# calls to SMT	91,000,000	1,600,000	2,000,000
Time (min)	492	7	10
Total calls to SMT	91,000,000	3,600,000	
Total time (min)	492	17	

25x less calls to the SMT solver (Yices) by reusing the SC boolean program

Thank you!

Questions?