

Context-Bounded Model Checking with ESBMC 1.17

Lucas Cordeiro, Jeremy Morse,
Denis Nicole, Bernd Fischer



UFAM

UNIVERSITY OF
Southampton
School of Electronics
and Computer Science

ESBMC: SMT-based BMC of single- and multi-threaded software

- exploits SMT solvers and its background theories to:
 - provide optimized encodings for pointers, bit operations, unions and arithmetic over- and underflow
 - efficient search methods (non-chronological backtracking, conflict clauses learning)
- supports verifying multi-threaded software that uses pthreads threading library
 - interleaves only at “visible” instructions
 - *lazy exploration* of the reachability tree
 - optional context-bound
- derived from CBMC

Lazy exploration of the Reachability Tree

Idea: iteratively generate all possible interleavings and call the BMC procedure on each interleaving

... combines

- **symbolic** model checking: on each individual interleaving
- **explicit state** model checking: explore all interleavings

Lazy exploration of the Reachability Tree

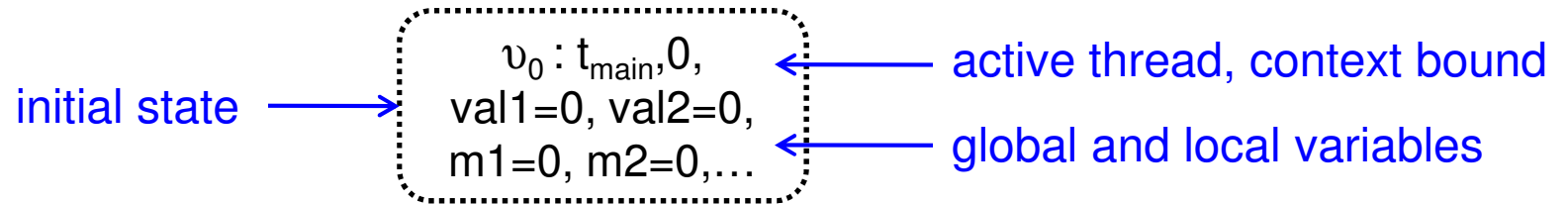
Lazy exploration of the Reachability Tree

initial state

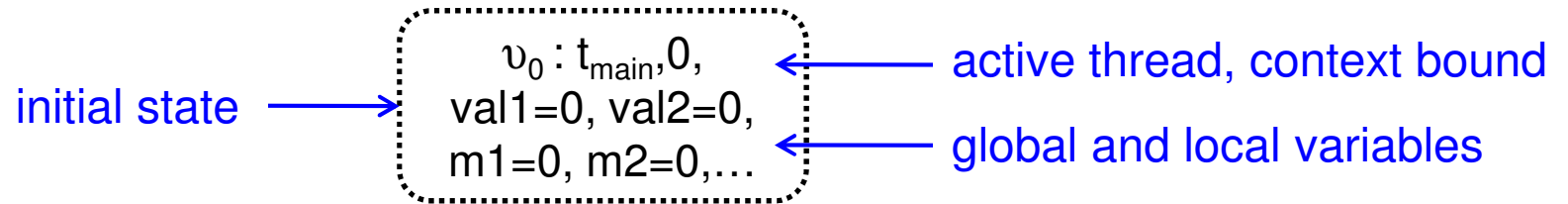


$v_0 : t_{\text{main}}, 0,$
 $\text{val1}=0, \text{val2}=0,$
 $m1=0, m2=0, \dots$

Lazy exploration of the Reachability Tree



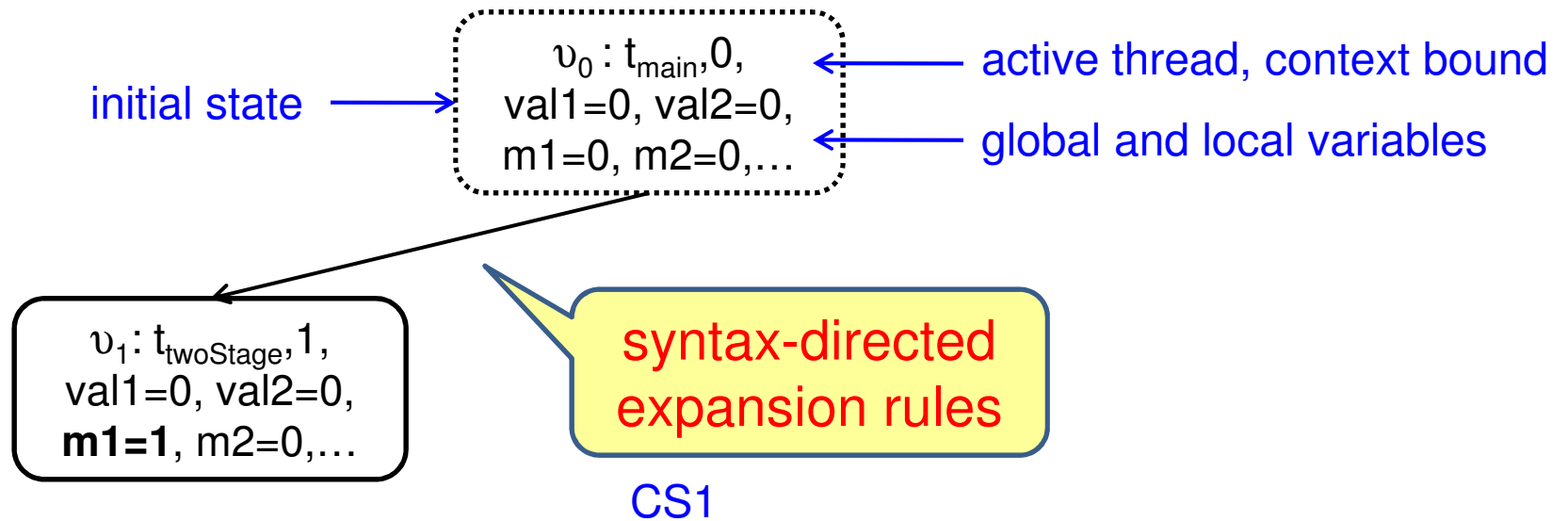
Lazy exploration of the Reachability Tree



CS1

CS2

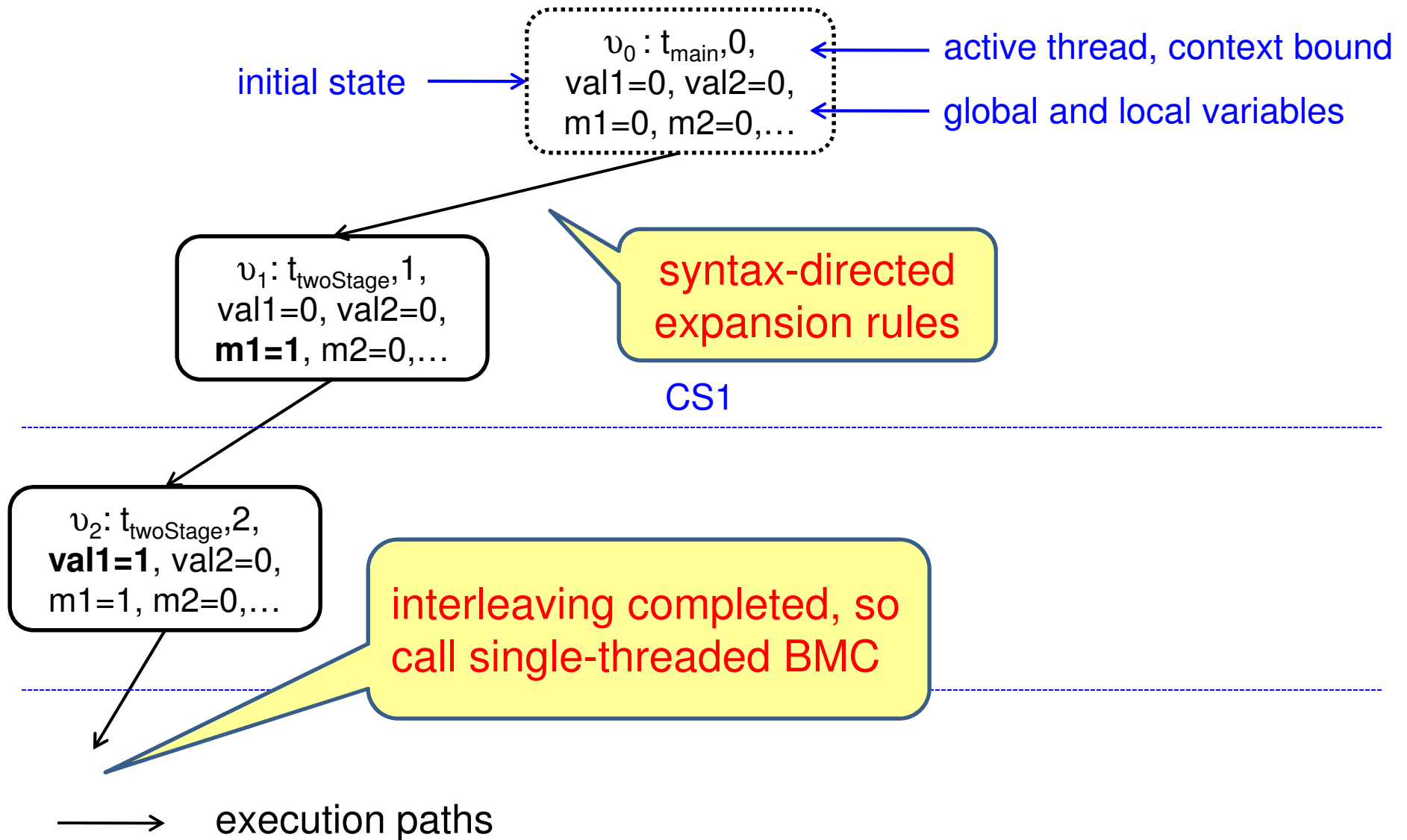
Lazy exploration of the Reachability Tree



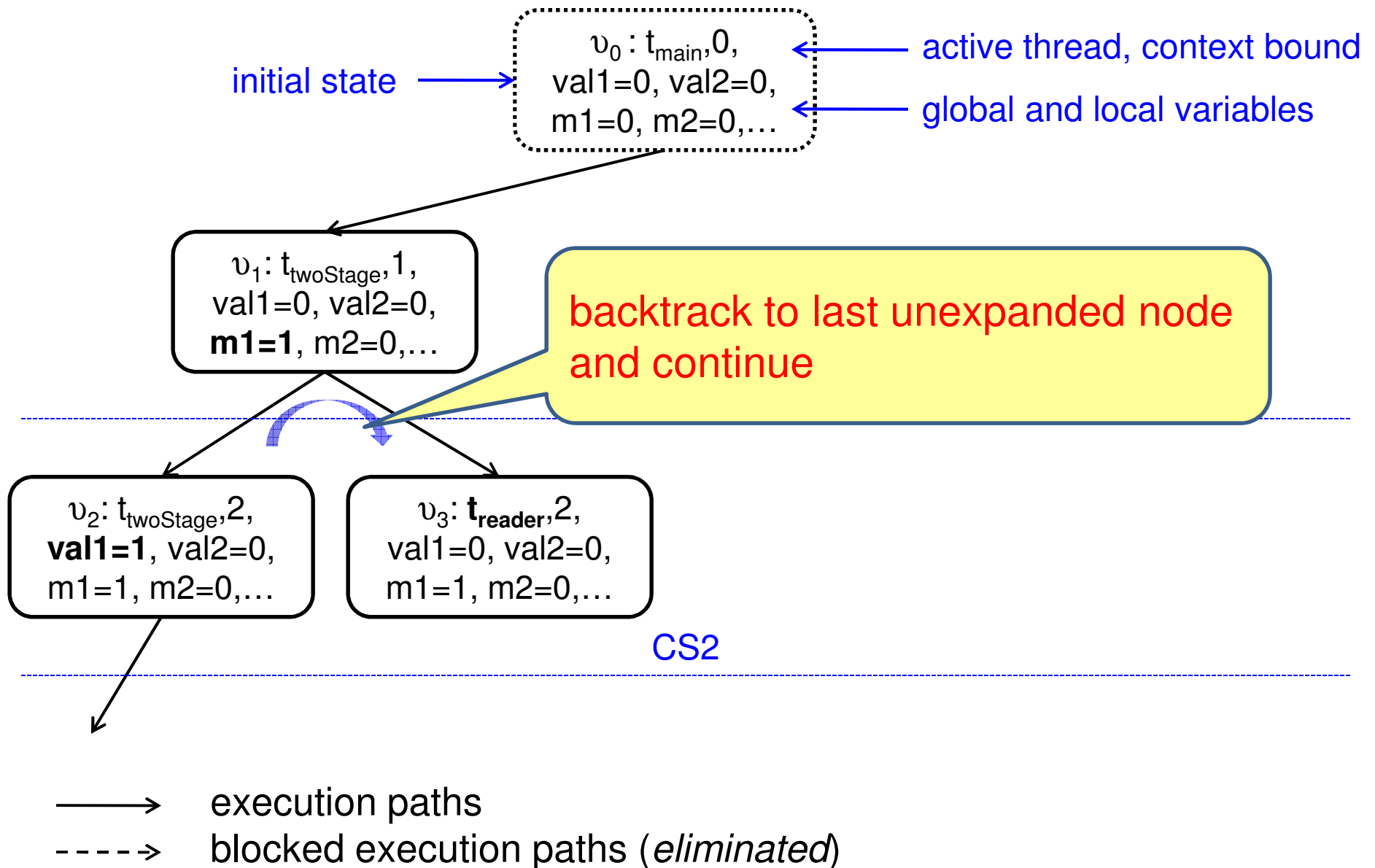
CS2

→ execution paths

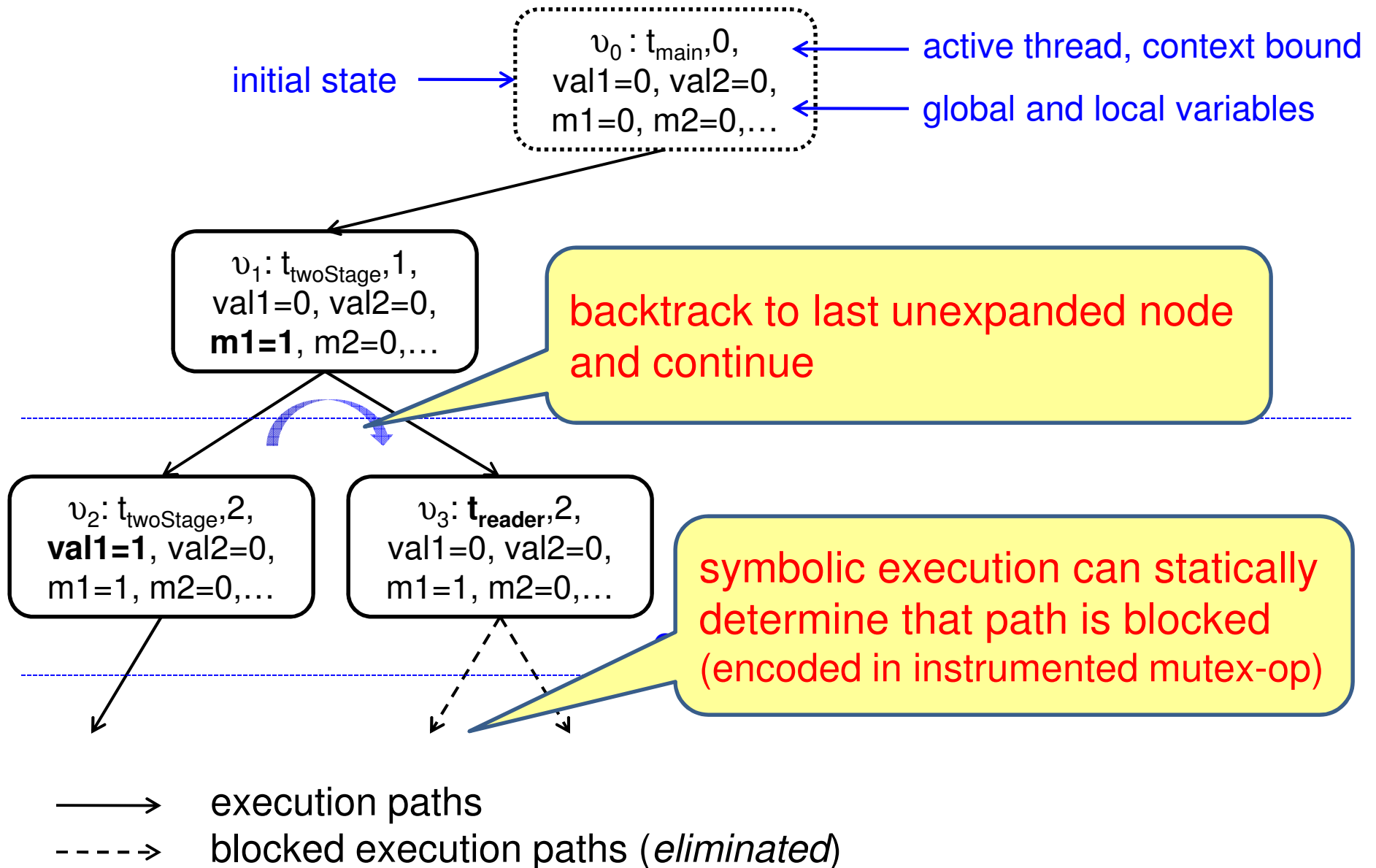
Lazy exploration of the Reachability Tree



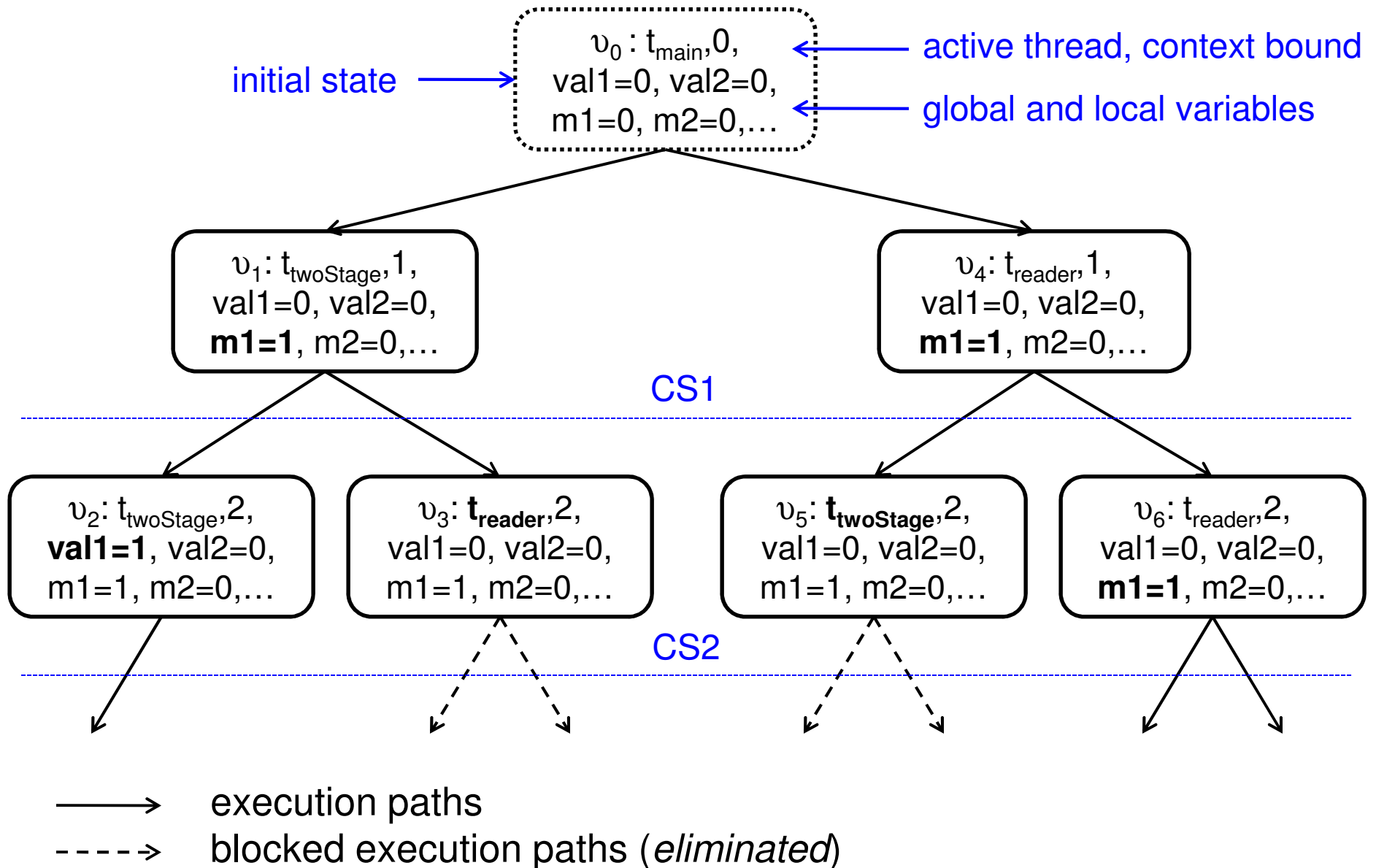
Lazy exploration of the Reachability Tree



Lazy exploration of the Reachability Tree



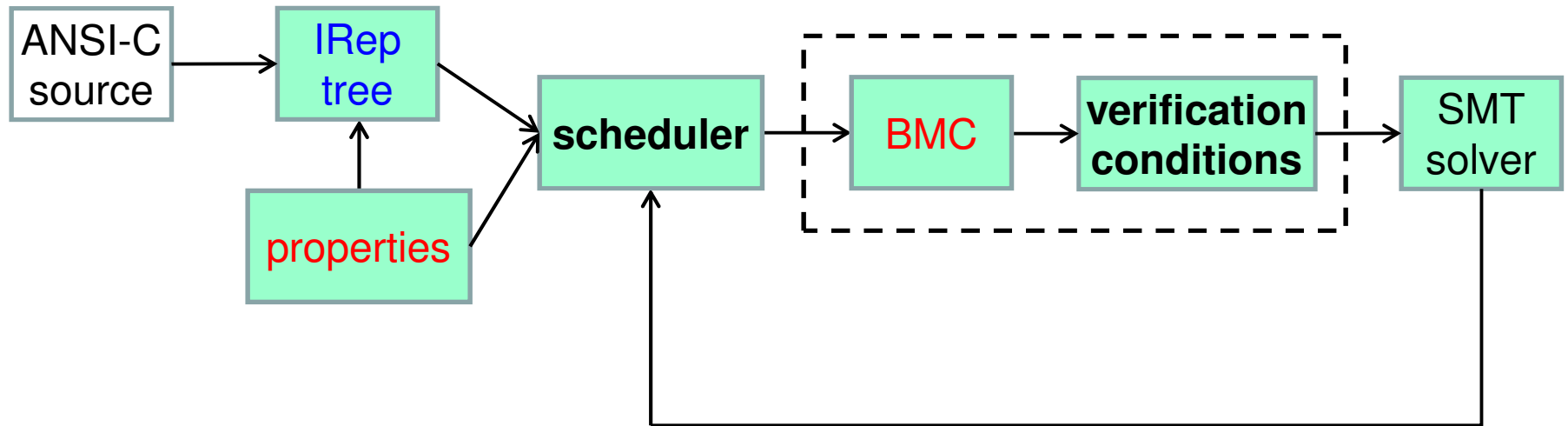
Lazy exploration of the Reachability Tree



ESBMC Verification Support

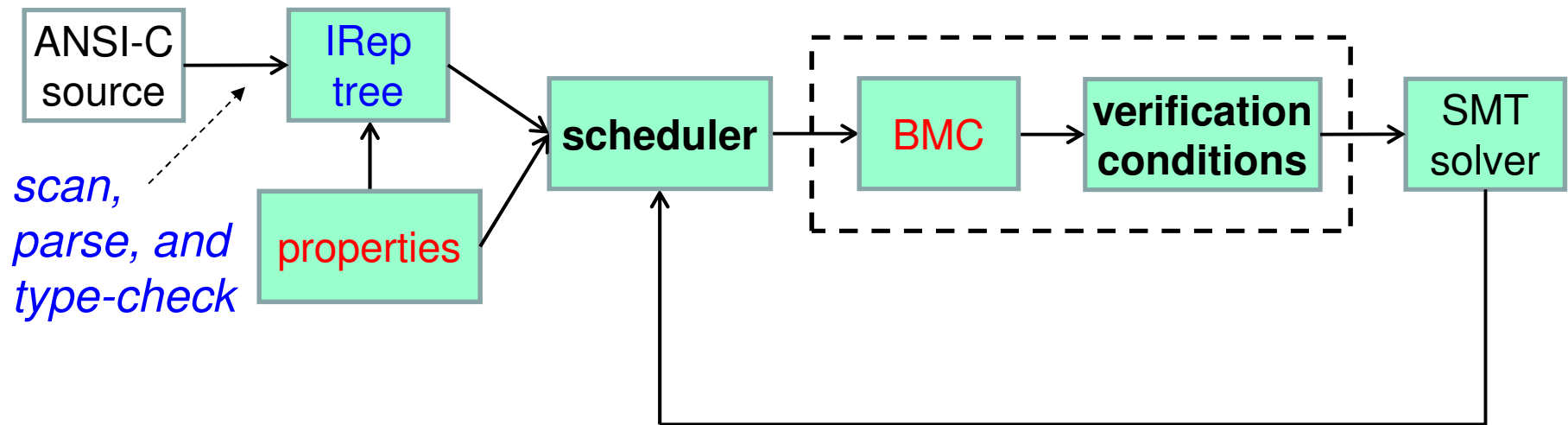
- built-in properties:
 - arithmetic under- and overflow
 - pointer safety
 - array bounds
 - division by zero
 - memory leaks
 - atomicity and order violations
 - deadlock
 - data race
- user-specified assertions
(*__ESBMC_assume*, *__ESBMC_assert*)
- built-in scheduling functions (*__ESBMC_atomic_begin*,
__ESBMC_atomic_end, *__ESBMC_yield*)

ESBMC Architecture



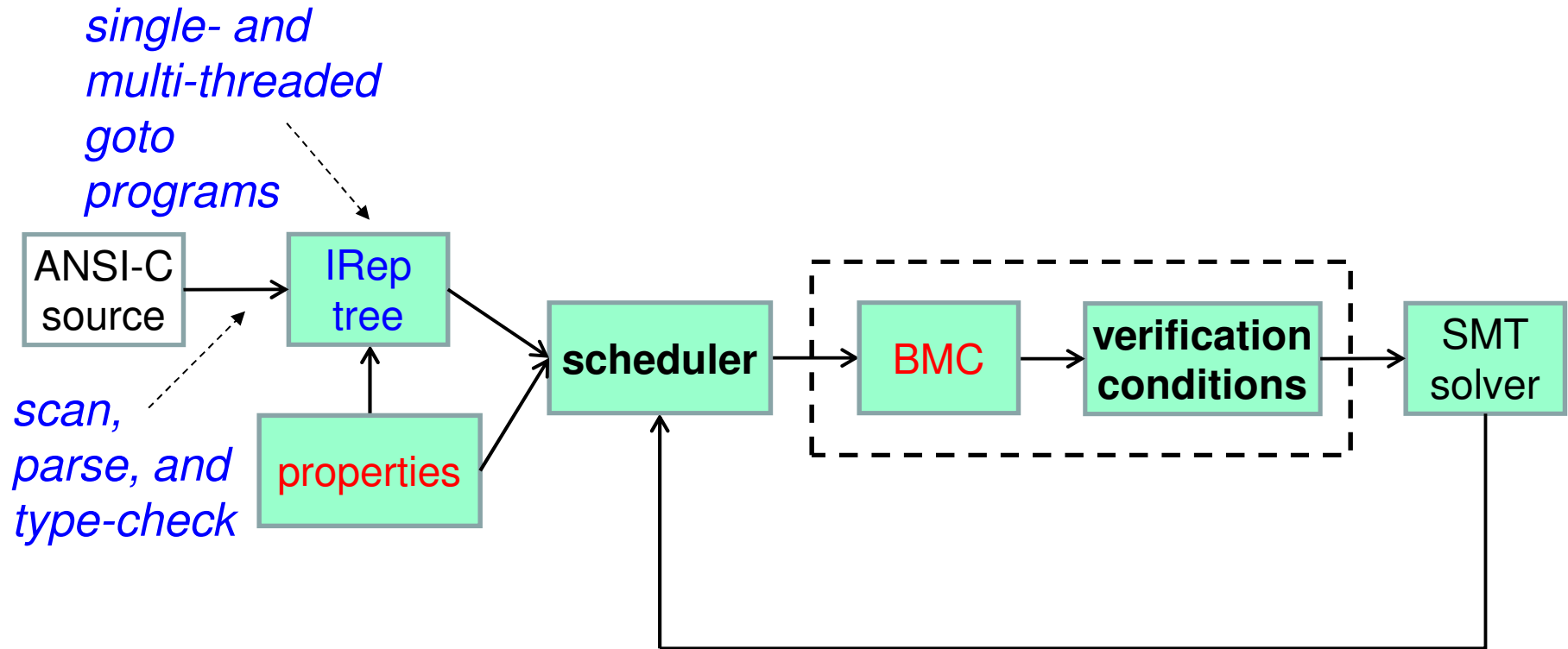
reused/extended from CBMC

ESBMC Architecture



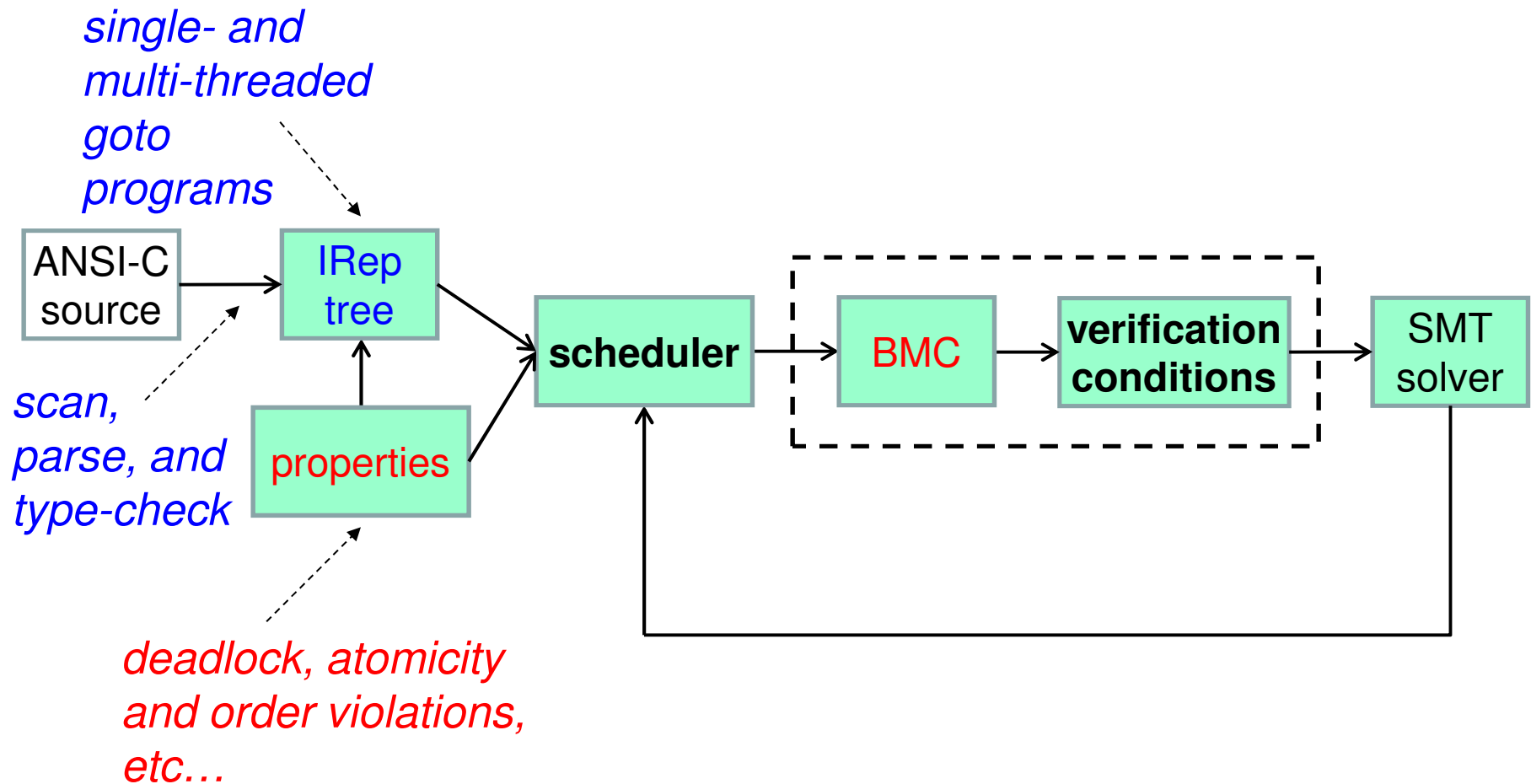
reused/extended from CBMC

ESBMC Architecture



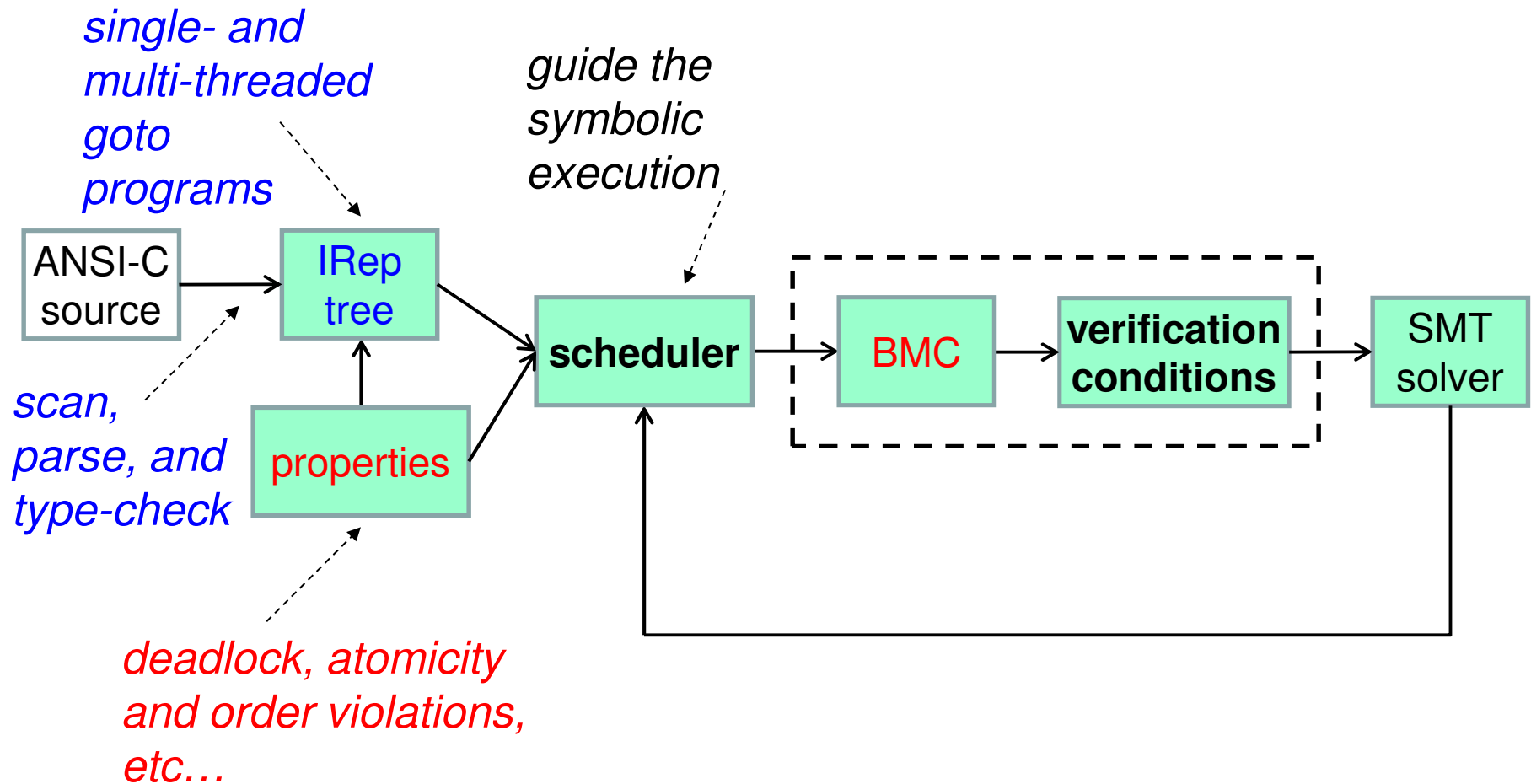
reused/extended from CBMC

ESBMC Architecture



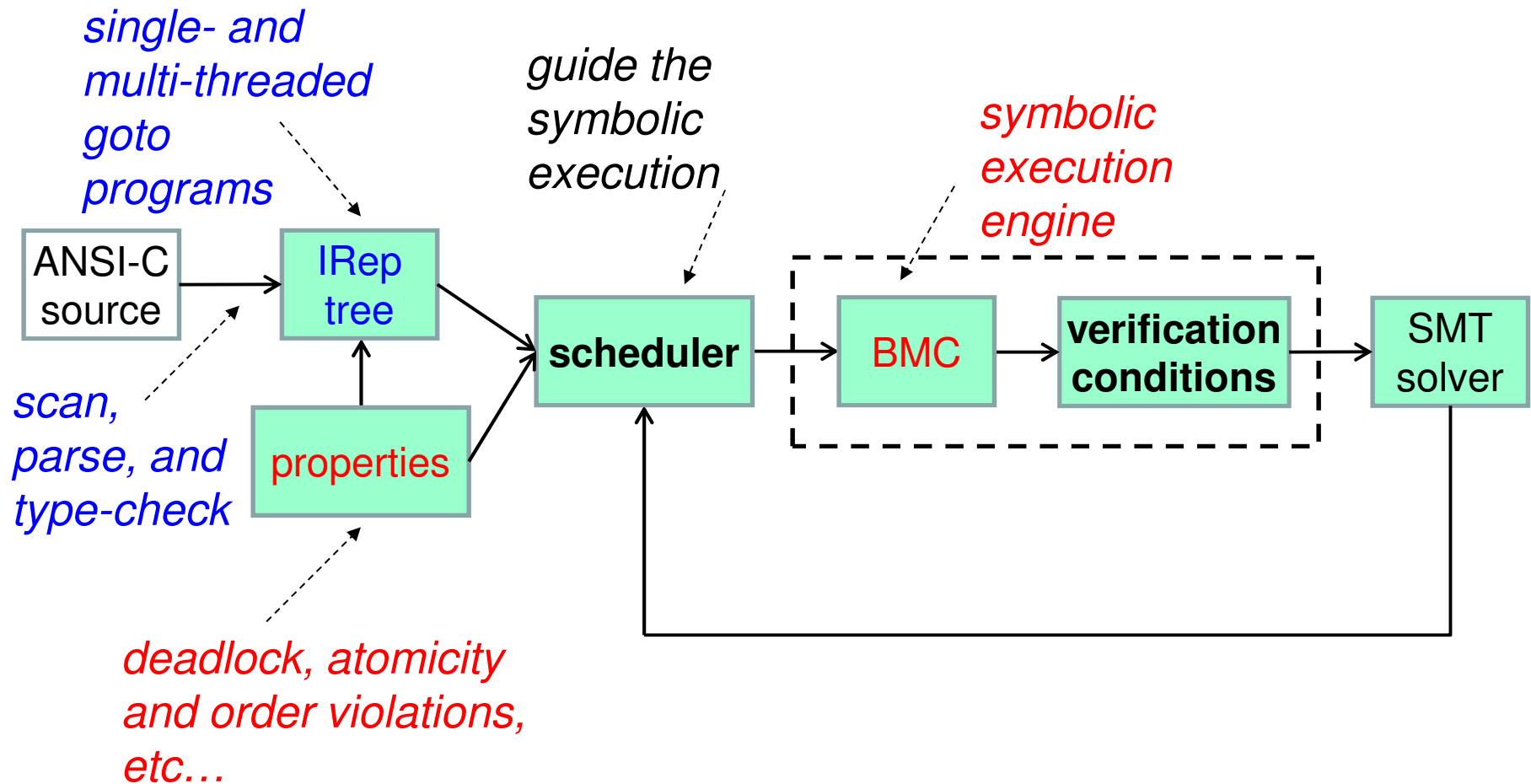
reused/extended from CBMC

ESBMC Architecture



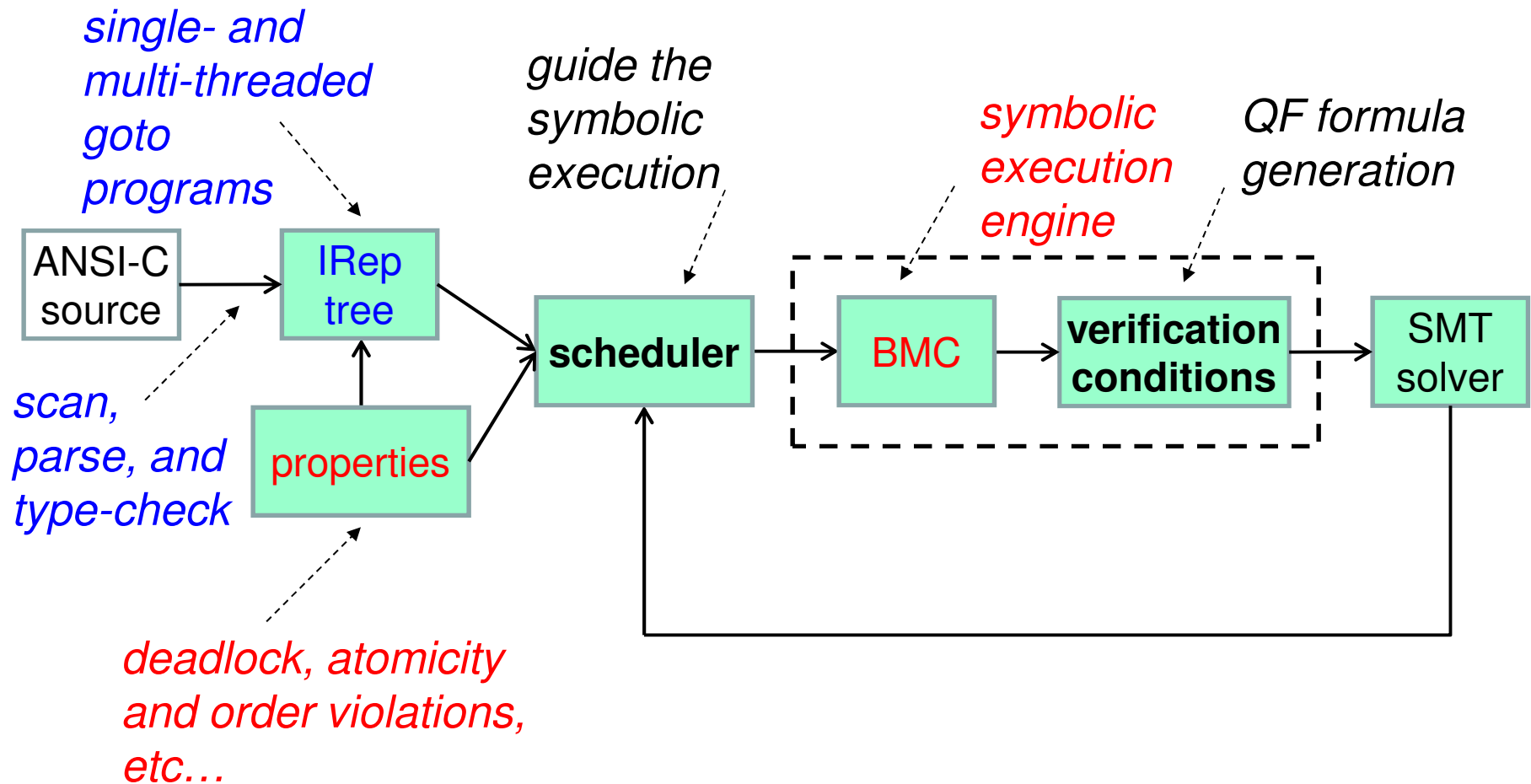
reused/extended from CBMC

ESBMC Architecture



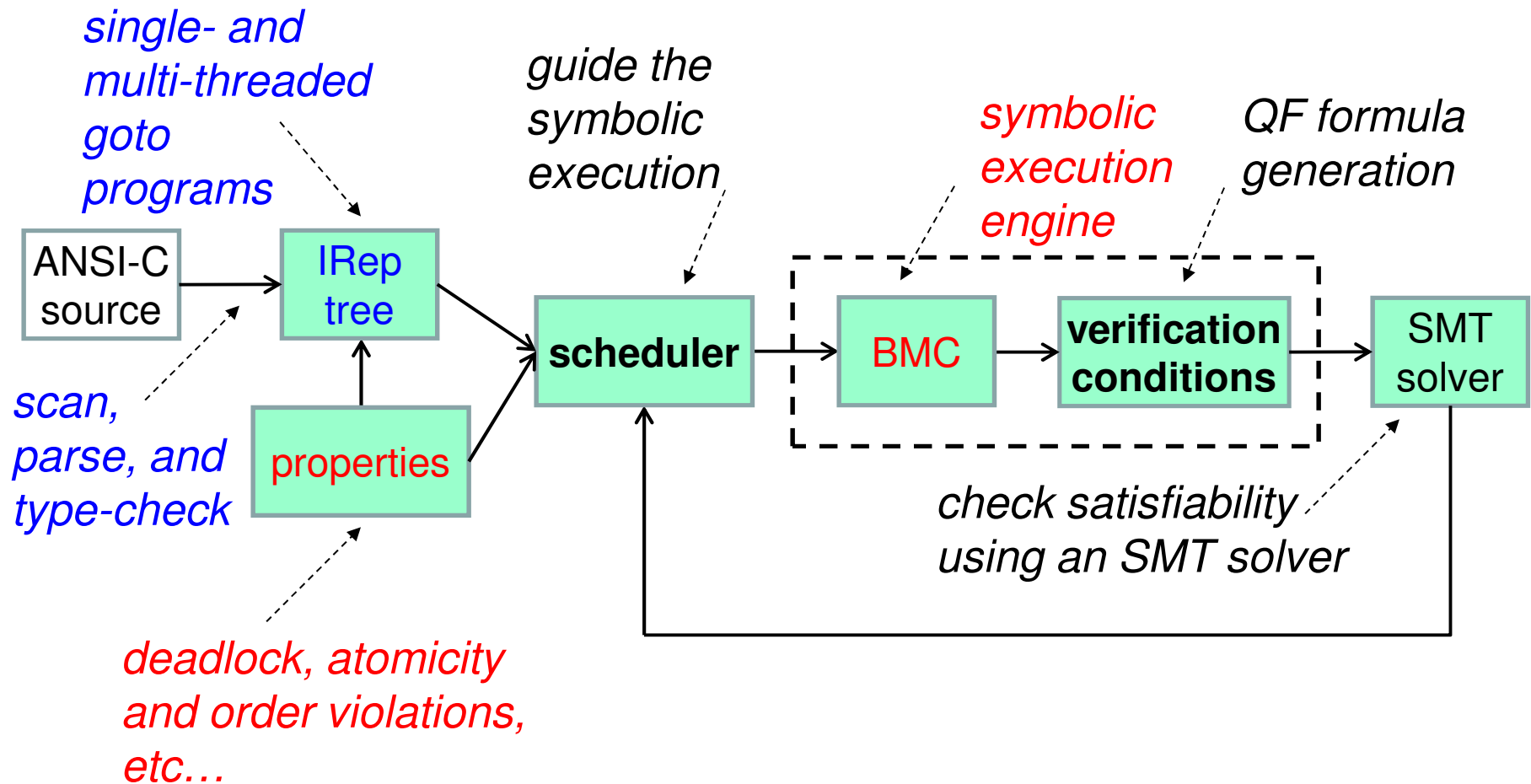
reused/extended from CBMC

ESBMC Architecture



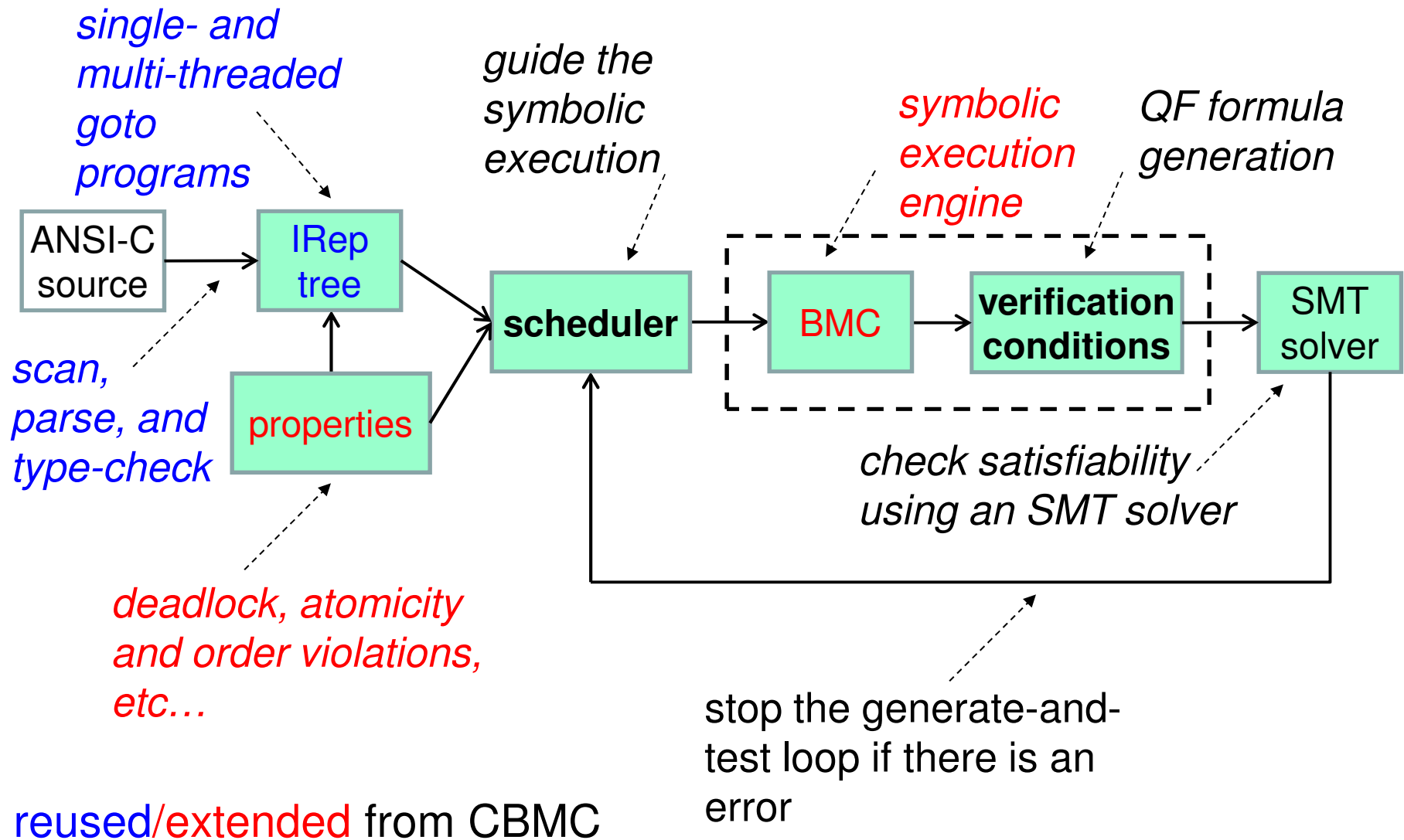
reused/extended from CBMC

ESBMC Architecture



reused/extended from CBMC

ESBMC Architecture



Strengths:

- robust context-bounded model checker for multi-threaded C code
- lazy exploration is fast for satisfiable instances and to a lesser extent even for safe programs
 - state hashing improves performance (modestly)

Strengths:

- robust context-bounded model checker for multi-threaded C code
- lazy exploration is fast for satisfiable instances and to a lesser extent even for safe programs
 - state hashing improves performance (modestly)

Weaknesses:

- scalability (like other BMCs...)
 - loop unrolling
 - interleavings
- pointer handling and points-to analysis
 - exposed by excessive typecasts in the CIL-converted code
 - better memory model in progress