Aalto University
School of Science

SMT Based State Reachability Checking for Multithreaded Programs

Kari Kähkönen and Keijo Heljanko

The Main Goal

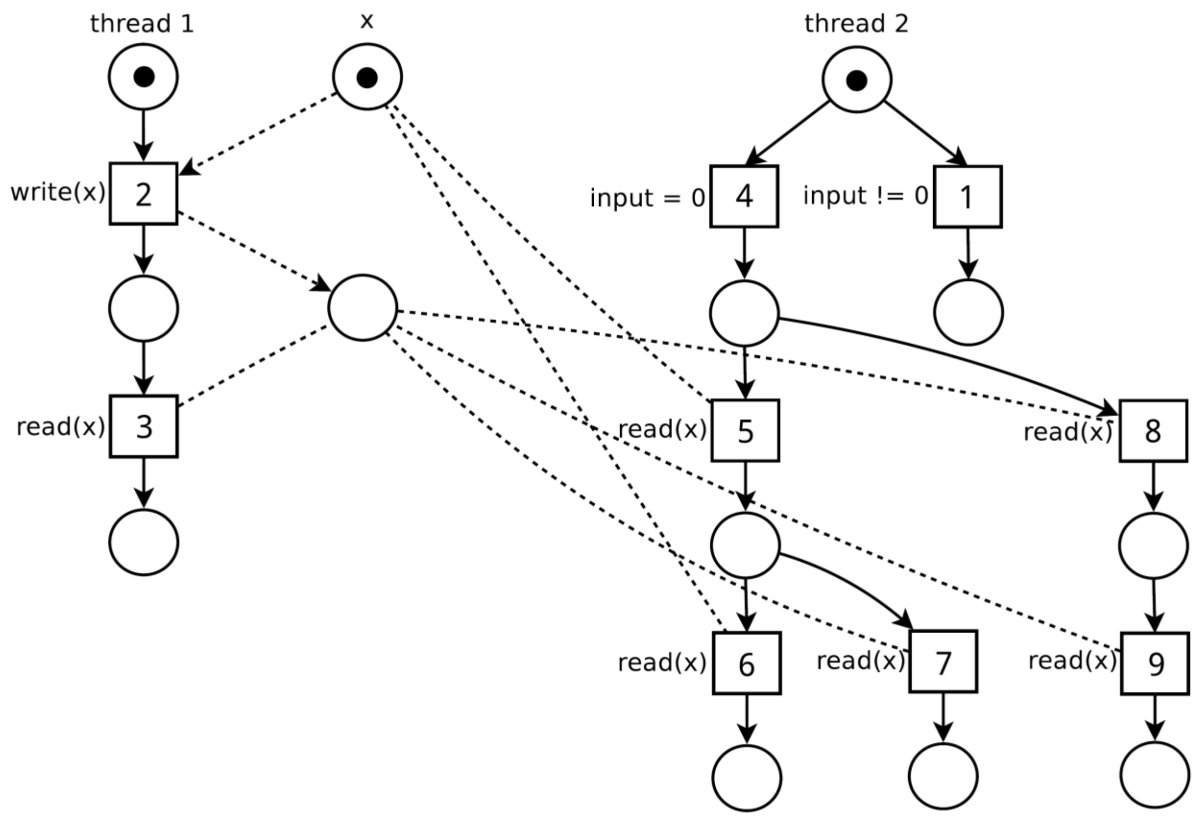
- Determine if a given global state is reachable in a multithreaded program
- More specifically: Given a set of test executions, can we predict that a given global state is reachable even if none of the test executions observed that state
- Our approach:
  - Model test executions as unfoldings (i.e., as a Petri net)
  - Translate the unfolding and the reachability problem into a SMT instance

Unfoldings of Multithreaded Programs

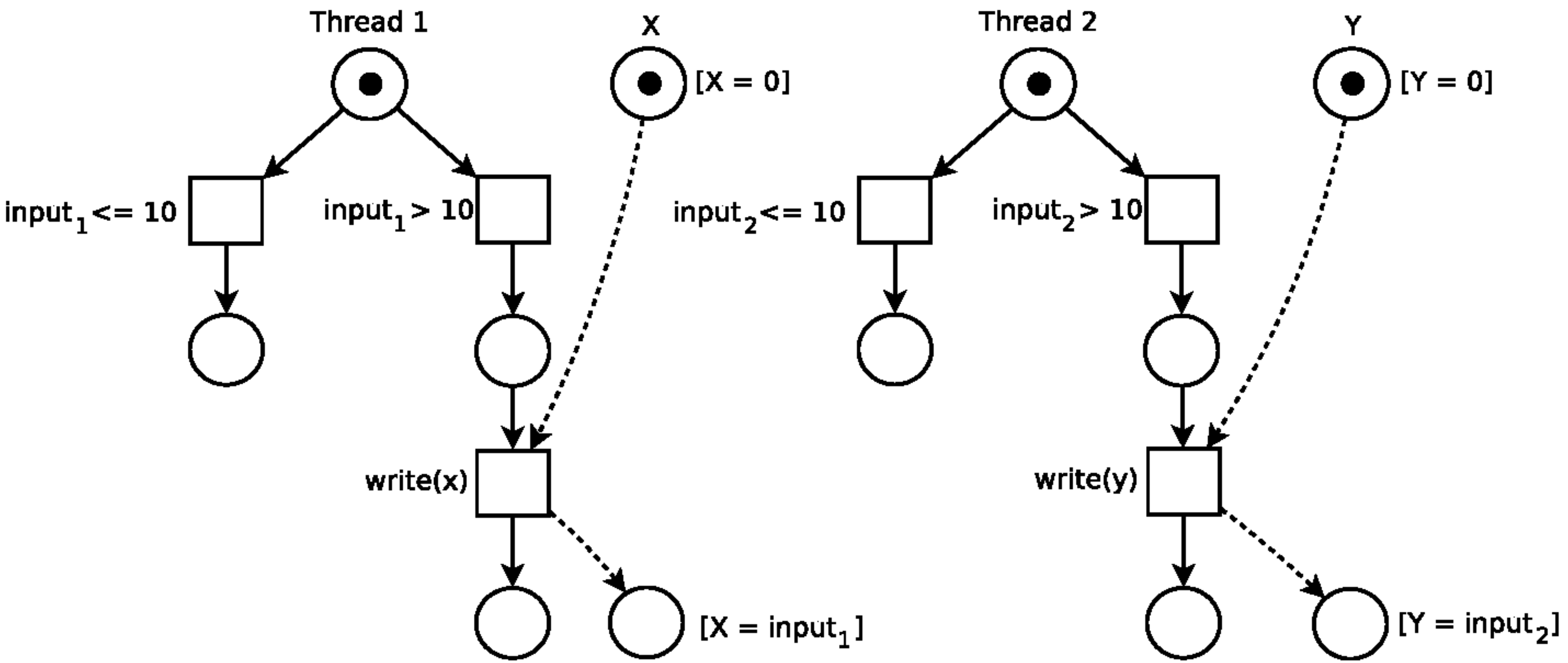
Global variables:
X = 0

Thread 1:
X = 5;
a = X;

Thread 2:
b = input();
if (b == 0)
    c = X;
    d = X;



Global State Reachability In Unfoldings



Is a global state satisfying X > 40 & Y = 15 reachable?

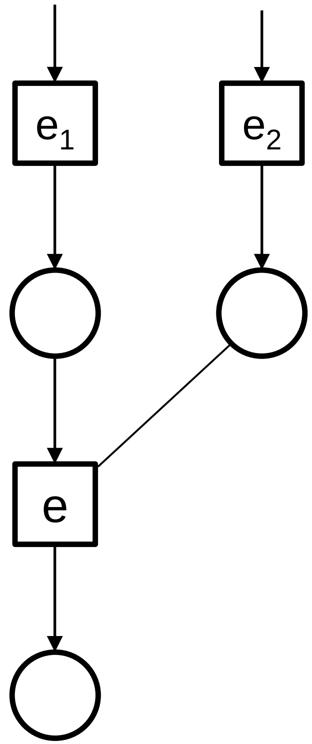
SMT Translation

- Each satisfying assignment is made to correspond to a reachable marking in the unfolding
  - Base translation captures all reachable markings
  - A global state property can then be added to the translation
- A boolean variable is created for each event and condition (i.e., for each transition and place)
- A variable for an event is true iff the event needs to be fired in order to reach the marking
- A variable for a condition is true iff it contains a token in the marking

SMT Translation (1)

For each event e

(1) $$e \Rightarrow \bigwedge_{e_i \in {}^\bullet({}^\bullet e) \cup {}^\bullet \underline{e}} e_i$$

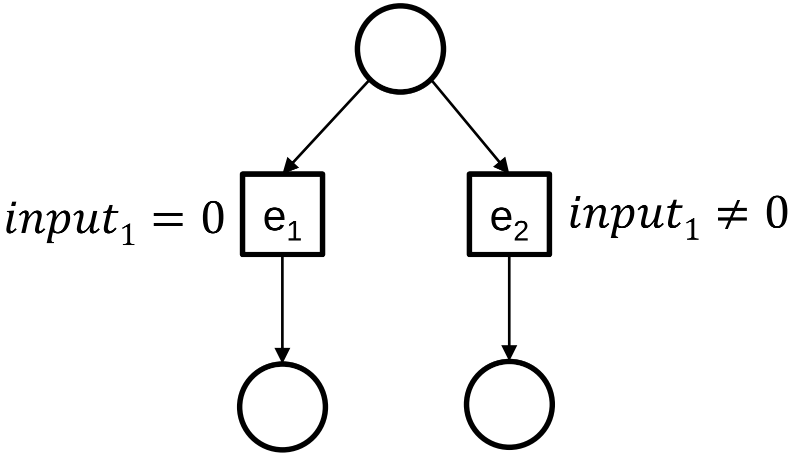


$$e \Rightarrow e_1 \wedge e_2$$

SMT Translation (2)

For each event e with a constraint g

(2) $$e \Rightarrow g$$



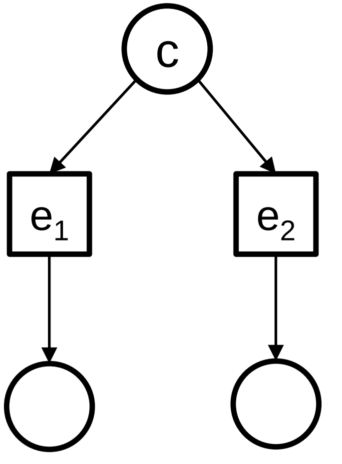
$$e_1 \Rightarrow input_1 = 0$$

$$e_2 \Rightarrow input_2 \neq 0$$

SMT Translation (3)

For each condition c and each event e in the postset of c

(3) $$e \Rightarrow \bigwedge_{e_i \in c^{\bullet} \setminus \{e\}} \neg e_i$$



$$e_1 \Rightarrow \neg e_2$$

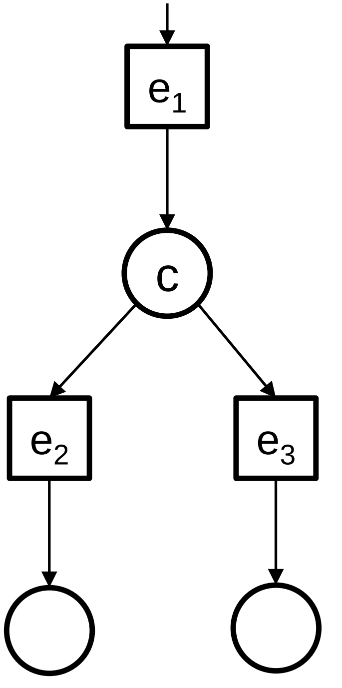
$$e_2 \Rightarrow \neg e_1$$

(Linear encoding is also possible)

SMT Translation (4)

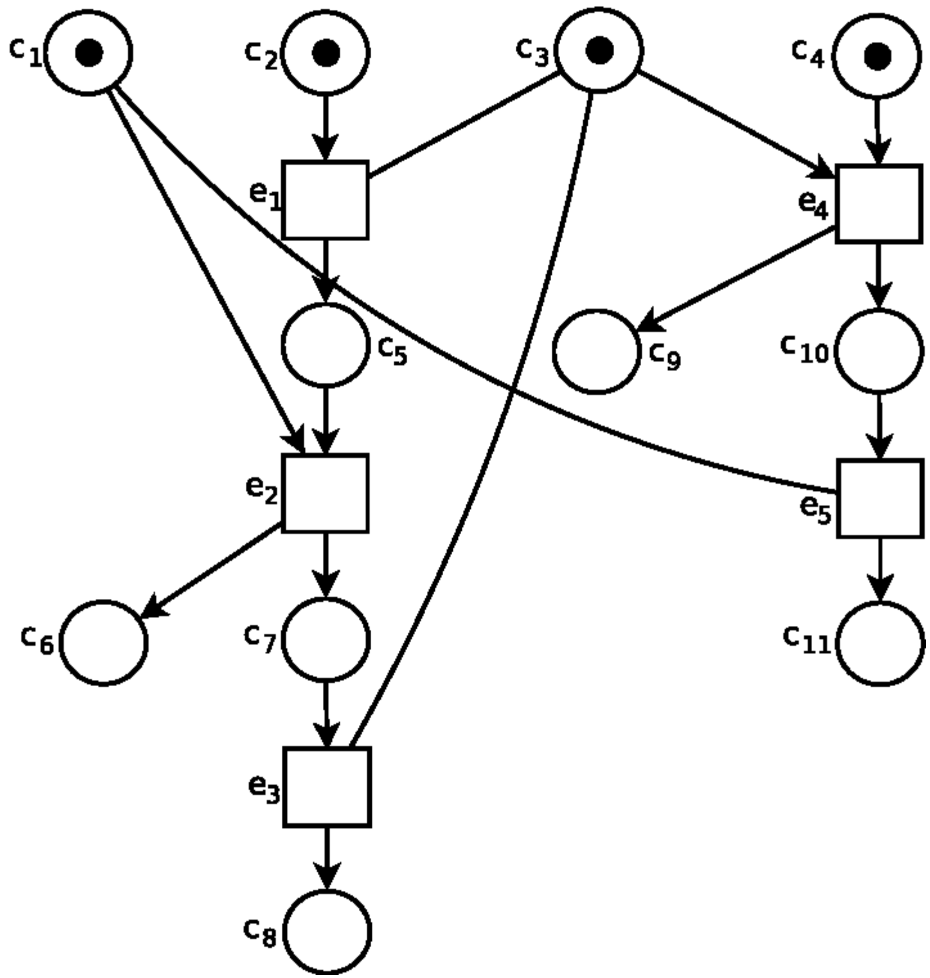
For each condition c and event e in the preset of c

(4) $$c \Rightarrow e \wedge \neg\left(\bigvee_{e_i \in c^\bullet} e_i\right)$$



$$c \Rightarrow e_1 \wedge \neg(e_2 \vee e_3)$$

Cycles of Asymmetric Conflicts



No reachable marking with both c_8 and c_{11}

e_5 must be fired before e_2
e_2 must be fired before e_3
e_3 must be fired before e_4
e_4 must be fired before e_5

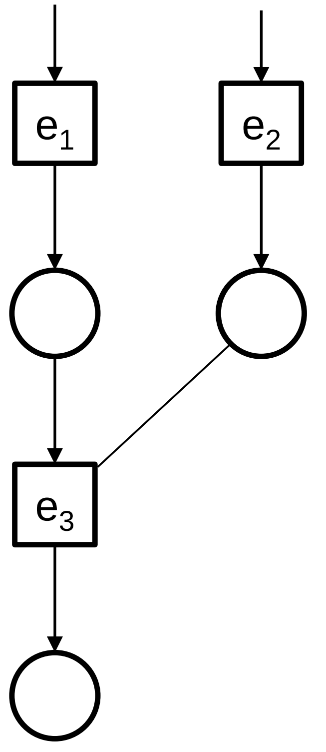
Handling Cycles in the SMT Translation

- We want the encoding for reachable markings to become unsatisfiable if the marking implies a cycle of asymmetric conflicts
- Idea: encode a valid firing order for the events
  - Create an interger variable describing this order for each event

SMT Translation (5)

For each event e_i and each event e_j in ${}^\bullet({}^\bullet e_i) \cup {}^\bullet e_j$

(5) $e_i \Rightarrow n_j < n_i$



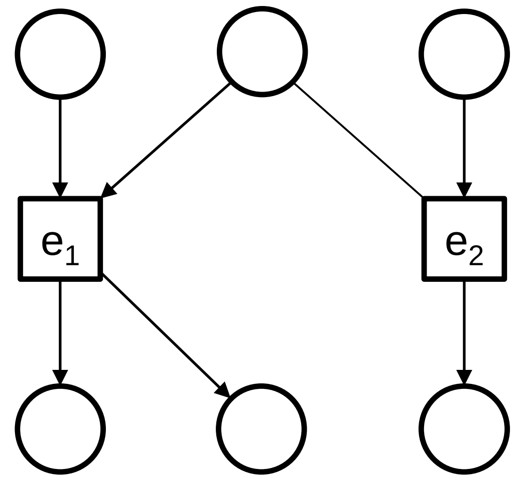
$e_3 \Rightarrow n_1 < n_3$

$e_3 \Rightarrow n_2 < n_3$

SMT Translation (6)

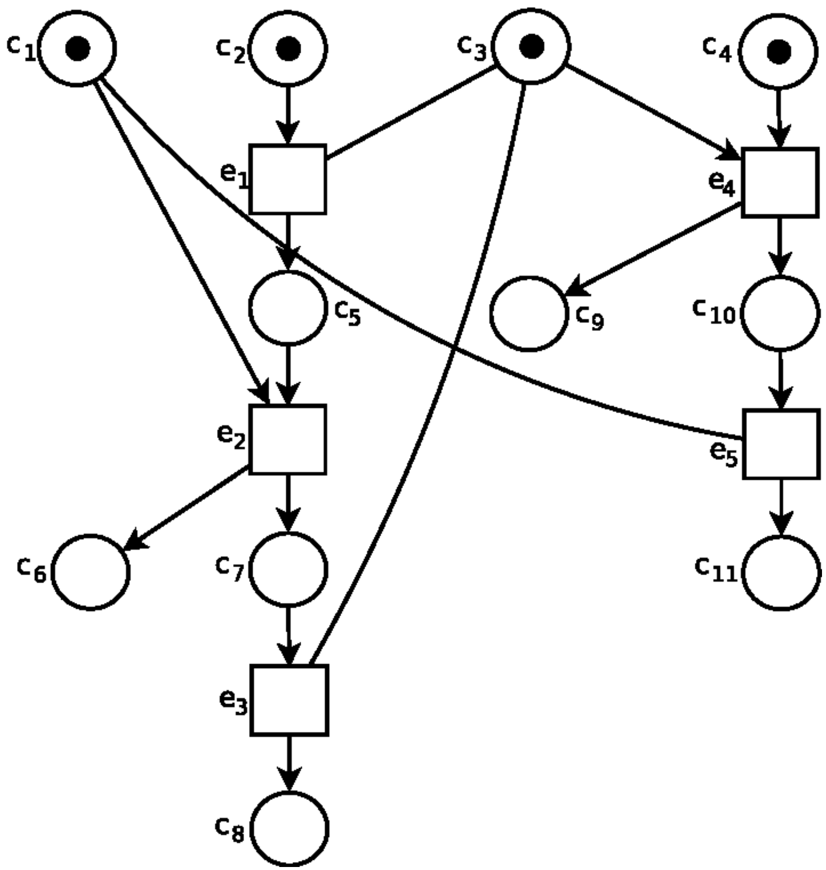
For each read event e_i and write event e_j that have a common condition in their context / preset

(6) $e_i \Rightarrow n_j < n_i$



$e_2 \Rightarrow n_2 < n_1$

Example



$e_2 \Rightarrow n_1 < n_2$
$e_3 \Rightarrow n_2 < n_3$
$e_5 \Rightarrow n_4 < n_5$

$e_1 \Rightarrow n_1 < n_4$
$e_3 \Rightarrow n_3 < n_4$
$e_5 \Rightarrow n_5 < n_2$

$e_2 \Rightarrow e_1$
$e_3 \Rightarrow e_2$
$e_5 \Rightarrow e_4$

$c_1 \Leftrightarrow \neg e_2$
$c_2 \Leftrightarrow \neg e_1$
$c_3 \Leftrightarrow \neg e_4$
$c_4 \Leftrightarrow \neg e_4$
$c_5 \Leftrightarrow e_1 \wedge \neg e_2$
$c_6 \Leftrightarrow e_2$
$c_7 \Leftrightarrow e_2 \wedge \neg e_3$
$c_8 \Leftrightarrow e_3$
$c_9 \Leftrightarrow e_4$
$c_{10} \Leftrightarrow e_4 \wedge \neg e_5$
$c_{11} \Leftrightarrow e_5$

Experiments

Benchmark	Property	SAT	Without read arcs	With read arcs
Updater	$x+y > 200 \wedge y < 100$	UNSAT	0m 49s	>30m
Updater	$x + y > 200$	SAT	0m 47s	>30m
Synthetic 3	$i+j = 50 \wedge k = -32 \wedge i >152$	SAT	2m 2s	0m 46s
Fib 1	$i \geq 32 \vee j \geq 32$	SAT	0m 41s	0m 12s
Fib 1	$i \geq 144 \vee j \geq 144$	UNSAT	0m 39s	0m 38s
Fib 2	$i \geq 32 \vee j \geq 32$	SAT	4m 28s	2m 29s
Fib 2	$i \geq 144 \vee j \geq 144$	SAT	7m 2s	26m 18s
Fib 2	$i > 144 \vee j > 144$	UNSAT	7m 15s	29m 54s

Conclusions

- Unfoldings of programs can be used to determine if a given global state is reachable in the program
- Global states can be searched directly from the unfolding or a SMT solver can be used as the search engine
- Unfoldings with read arcs can contain cycles of asymmetric conflicts
  - Makes the SMT translation more demanding to solve
  - Perhaps there is a better way to handle the cycles?