Introduction to Software Verification

Vincent Penelle

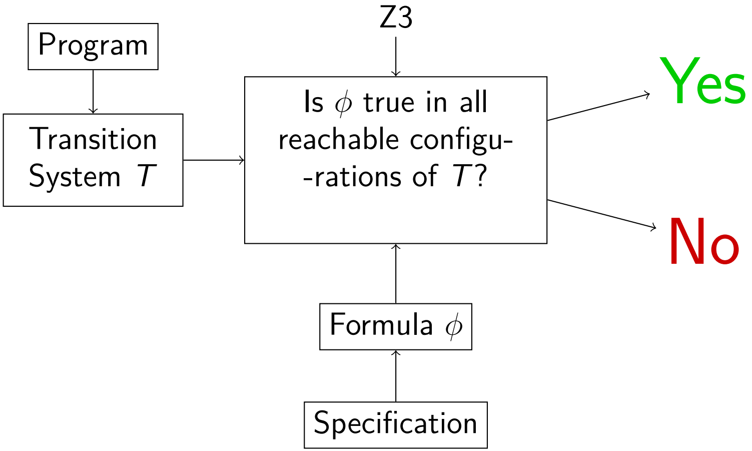
<vpenelle@u-bordeaux.fr>

LaBRI, Université de Bordeaux

September 23, 2019

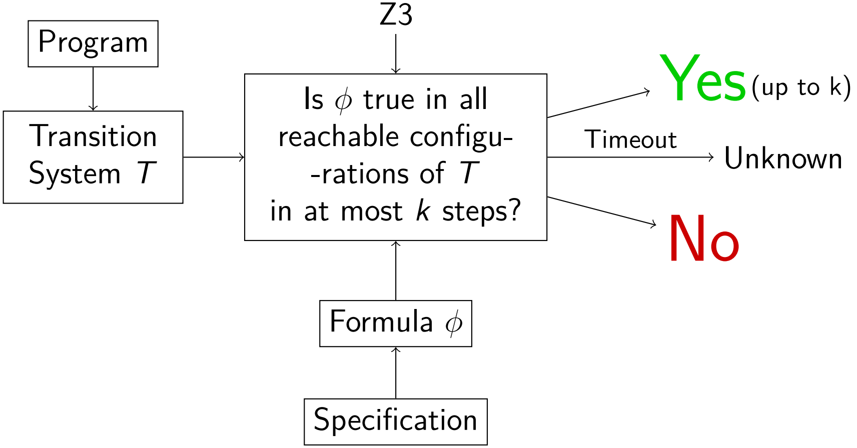
— *Bounded Model Checking* —

Model-checking



T is a priori infinite! So it is not possible to give to a SMT solver like that!

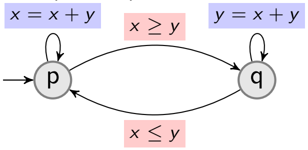
Bounded Model-checking



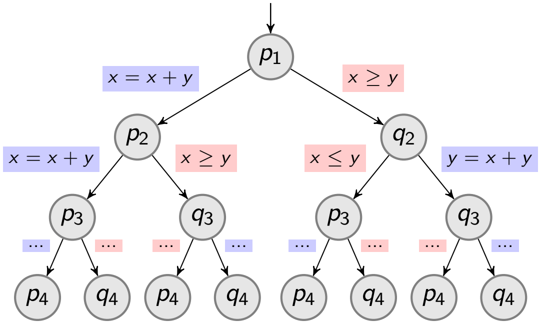
The unfolding of T up to k step is finite! Therefore Z3 can test if T holds up to k. Of course, we can increase k if the answer is yes and we have a doubt.

Unfolding a transition system

A (simple) system.

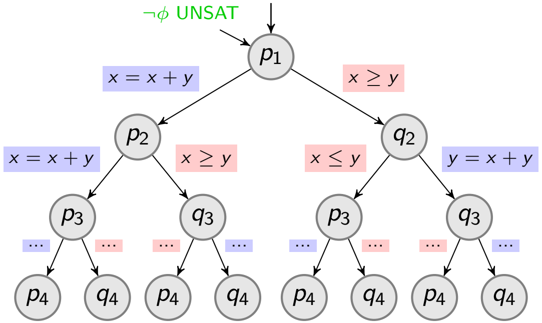


Its unfolding up to 4.



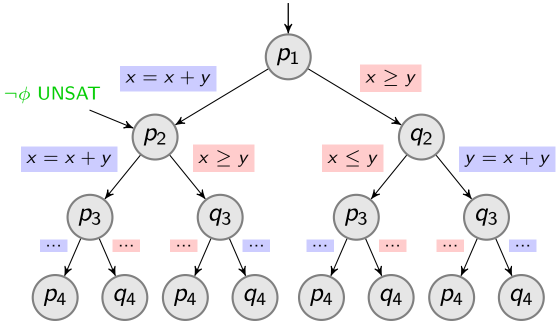
Depth-first search

Given a formula ϕ, we want to know if it is **valid** (it is the specification) up to depth k in all executions. We perform a **depth-first** search.



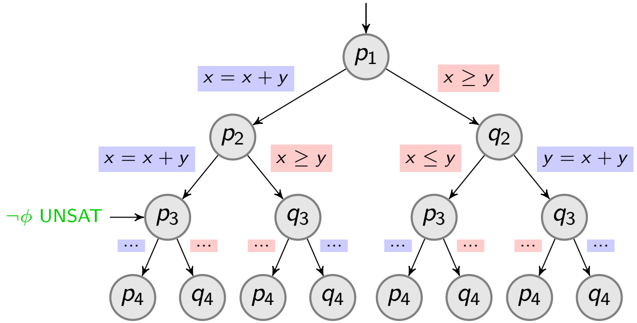
Depth-first search

Given a formula ϕ, we want to know if it is **valid** (it is the specification) up to depth k in all executions. We perform a **depth-first** search.



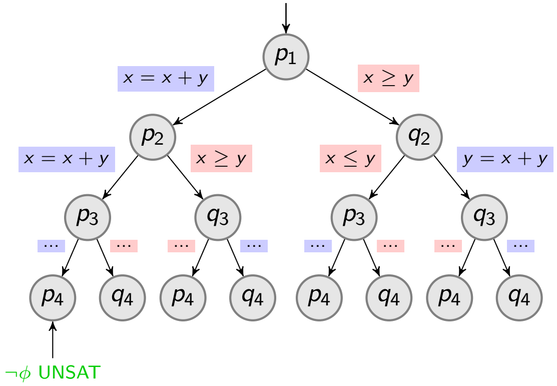
Depth-first search

Given a formula ϕ, we want to know if it is **valid** (it is the specification) up to depth k in all executions. We perform a **depth-first** search.



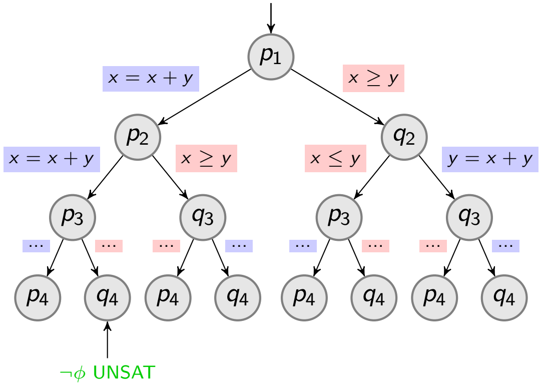
Depth-first search

Given a formula ϕ, we want to know if it is **valid** (it is the specification) up to depth k in all executions. We perform a **depth-first** search.



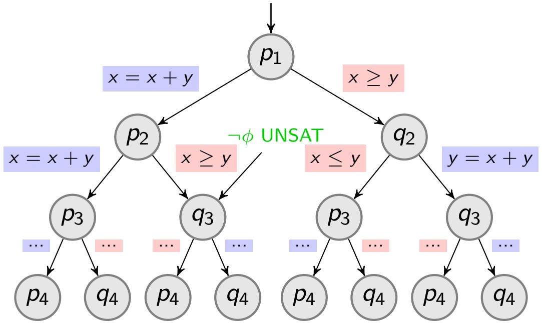
Depth-first search

Given a formula ϕ, we want to know if it is **valid** (it is the specification) up to depth k in all executions. We perform a **depth-first** search.



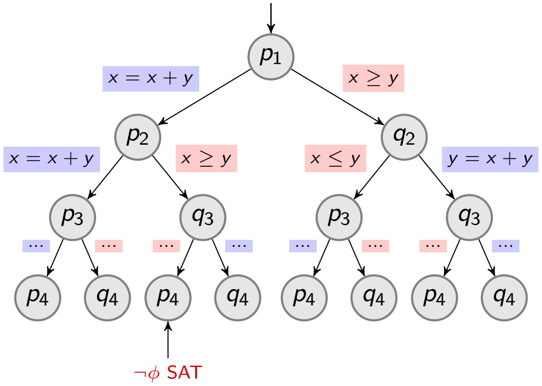
Depth-first search

Given a formula ϕ, we want to know if it is **valid** (it is the specification) up to depth k in all executions. We perform a **depth-first** search.



Depth-first search

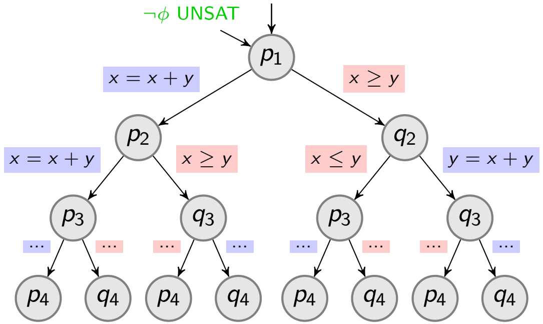
Given a formula ϕ, we want to know if it is **valid** (it is the specification) up to depth k in all executions. We perform a **depth-first** search.



We know ϕ can be falsified with the execution p_1, p_2, q_3, p_4. It is the **trace of error** and we output it!

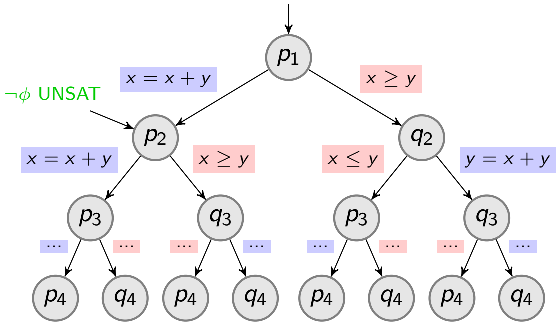
Breadth-first search

Another solution to do bounded model-checking is to perform a **breadth-first** search.



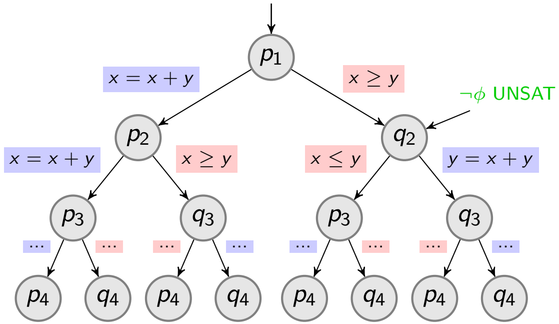
Breadth-first search

Another solution to do bounded model-checking is to perform a **breadth-first** search.



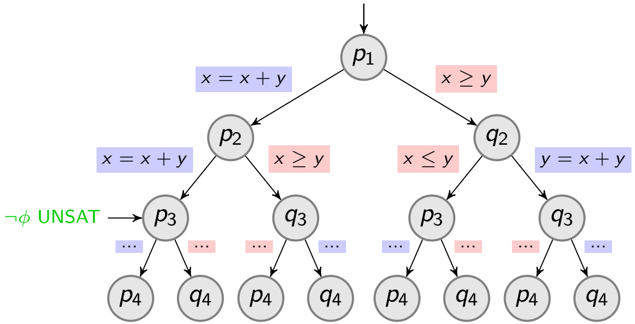
Breadth-first search

Another solution to do bounded model-checking is to perform a **breadth-first** search.



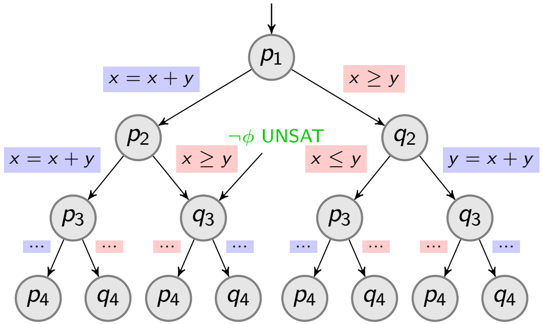
Breadth-first search

Another solution to do bounded model-checking is to perform a **breadth-first** search.



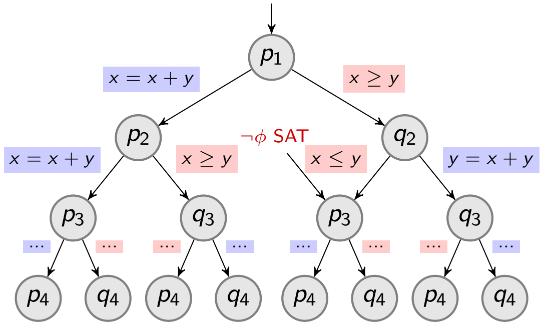
Breadth-first search

Another solution to do bounded model-checking is to perform a **breadth-first** search.



Breadth-first search

Another solution to do bounded model-checking is to perform a **breadth-first** search.

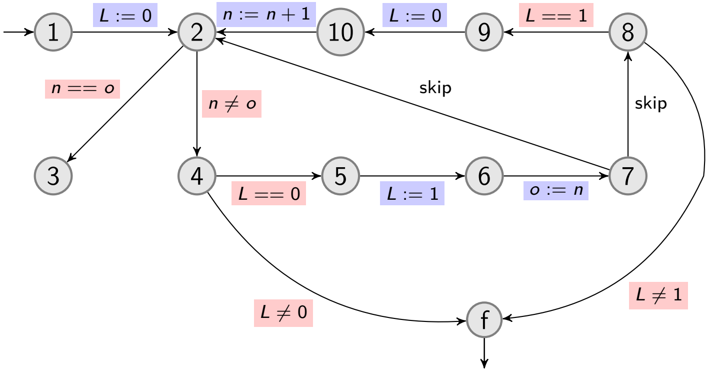


Here, p_1, q_2, p_3 is the **trace of error** and we output it!

Forward versus backward

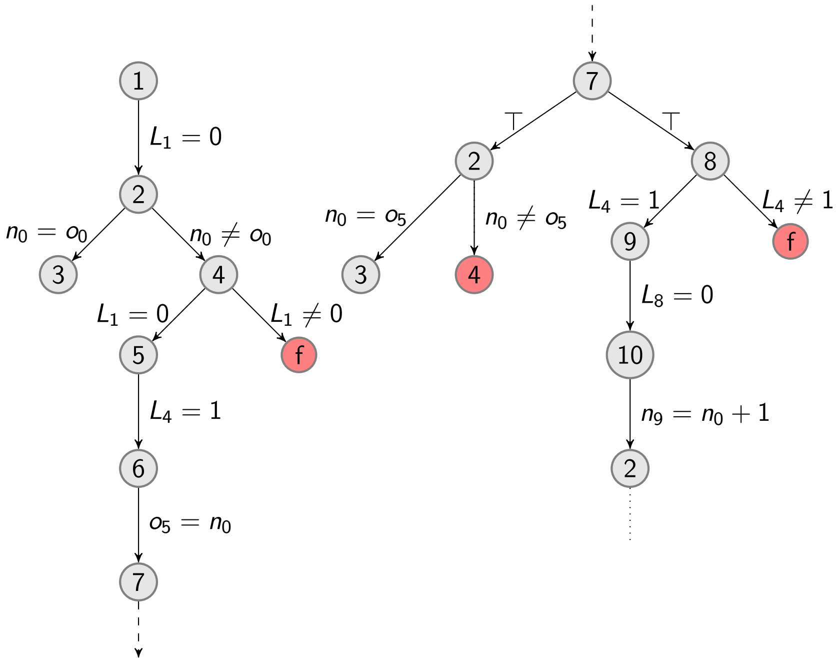
Forward: unfold the transition system in forward direction (as the execution is actually done).
Backward: unfold the transition system in reverse direction.
As logical formula are not concerned with the direction of execution, the two version are equivalent. There are cases in which one is better than the other, and some where it doesn't matter.

Forward versus bacward: an example

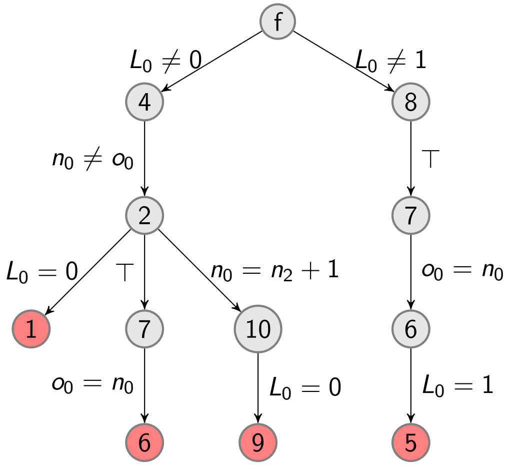


For forward bounded model-checking, we get a non-exhaustive search (fail not reachable).
For backward model-checking, we get an exhaustive search (depth 5).

Forward version



Bacward version



Completeness threshold

Knowing when we know we have a reliable answer:
As hard as MC to compute exactly.
Possible to over-approximate it by seeing the system as a graph.

Correctness

Détecter les boucles ? Faut que je discute de ça. . .

Example

Wolf, cabbage, goat, boat.