Handling State Space Explosion in Model Checking

ECS289 Project by Jannik Hiller

What even is Model Checking?

Made by ChatGPT

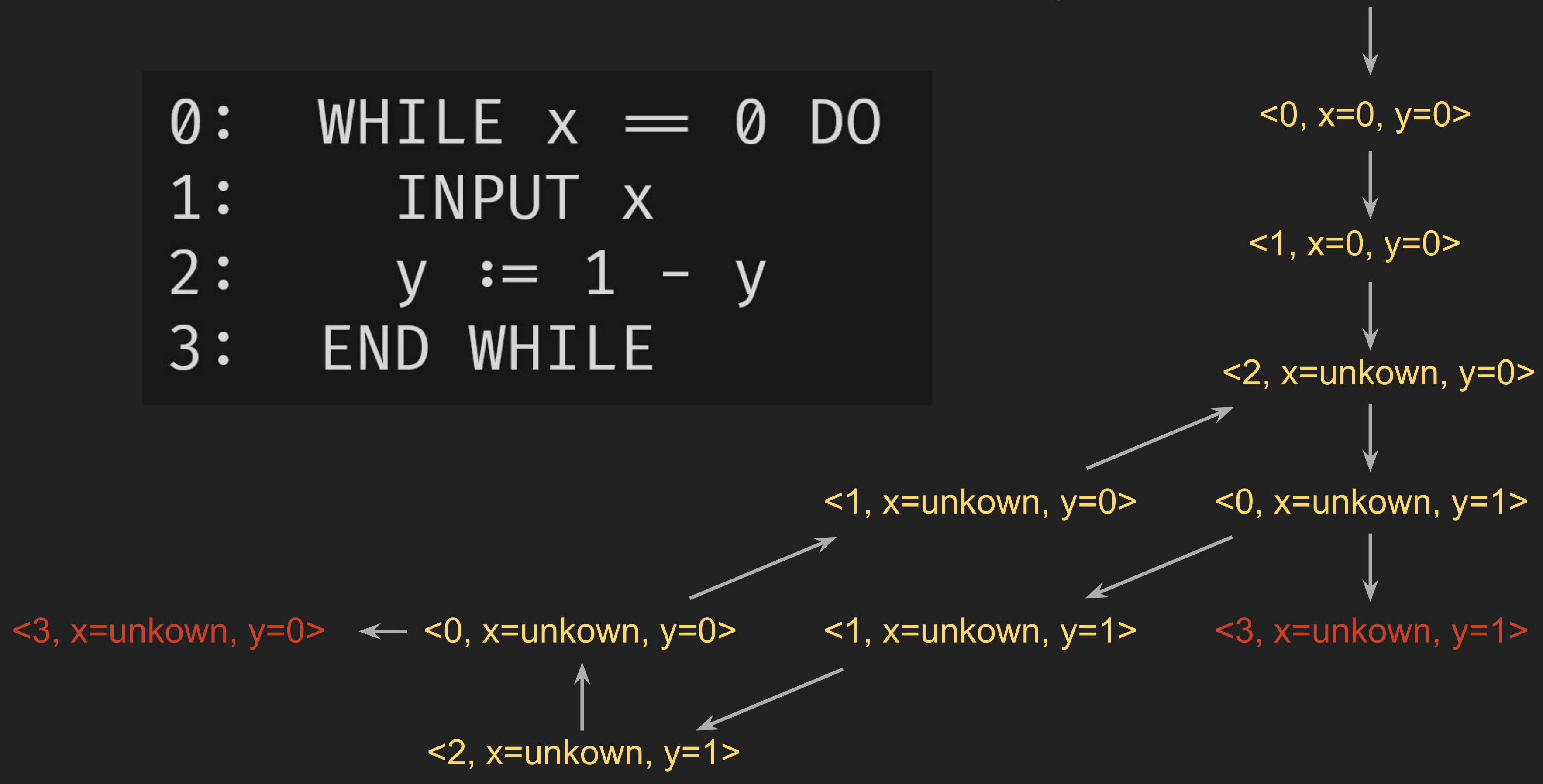
Short Introduction to Model Checking

- Methodology of verifying specifications of a ***Finite State Model*** (Transition System / Kripke Structure)
- A node in a transition system can represent:
  - the state of transistors in an integrated circuit
  - the state of an aircraft control system
  - the ***state of a program*** at any point in execution

- The state of a program: Current Instruction + Values of all variables

Example of a Transition System

```
0:   WHILE x == 0 DO
1:     INPUT x
2:     y := 1 - y
3:   END WHILE
```



State Space Explosion

Transition systems can be exponential in the program size!

```
INPUT x
INPUT y

WHILE a < x DO
  a := a + 1

  WHILE b < y DO
    b := b + 1

  END WHILE
END WHILE
```

685 states reachable in 50 steps

3535 states reachable in 50 steps

```
INPUT x
INPUT y
INPUT z

WHILE a < x DO
  a := a + 1

  WHILE b < y DO
    b := b + 1

    WHILE c < z DO
      c := c + 1

    END WHILE
  END WHILE
END WHILE
```

SAT based encoding[1]

We can't store all transitions directly => Represent them symbolically

Key Idea:

1. View states as vector of boolean values: <4, a=1, b=3> => 100 001 011
2. Consider variables $x = x_0x_1x_2\ \ x_3x_4x_5\ \ x_6x_7x_8$ and $y = y_0y_1y_2\ \ y_3y_4y_5\ \ y_6y_7y_8$
3. Create a boolean formula $T(x, y)$ that is true iff. there is a transition from x to y

We can create formulas that work for entire groups of transitions.
E.g. create a formula for each location in the source code.
Then $T(x, y)$ is the disjunction of these formulas.

[1]Biere, et al. 1999. Symbolic model checking without BDDs

SAT based encoding[1]

$x = x_0x_1x_2\ \ x_3x_4x_5\ \ x_6x_7x_8$ and $y = y_0y_1y_2\ \ y_3y_4y_5\ \ y_6y_7y_8$
T(x, y) true iff. there is a transition from x to y

```
3:    ...
4:   a := a + 1
5:    ...
```

$$T(x, y) \;=\; ((x_0 \Leftrightarrow 1 \wedge x_1 \Leftrightarrow 0 \wedge x_2 \Leftrightarrow 0) \wedge \quad (y_0 \Leftrightarrow 1 \wedge y_1 \Leftrightarrow 0 \wedge y_2 \Leftrightarrow 1)$$

4: → 5:

$$\wedge \quad (x_3 \oplus c_3 \Leftrightarrow y_3) \;\wedge\; (x_4 \oplus c_4 \Leftrightarrow y_4) \;\wedge\; (x_5 \oplus 1 \Leftrightarrow y_5)$$

$a_y = a_x + 1$

$$\wedge \quad (x_6 \Leftrightarrow y_7 \;\wedge\; x_7 \Leftrightarrow y_7 \;\wedge\; x_8 \Leftrightarrow y_8))$$

$b_y = b_x$

$$\vee \quad \ldots$$

(c_3 and c_4 are carry bits)

[1]Biere, et al. 2009. Symbolic model checking without BDDs

Made by ChatGPT

SAT based encoding[1]

- NP-hard, but in practice fast to solve
- Only one Conjunction per instruction

  => Encoding is linear in the program size
- All operations need to be implemented as boolean formulas ("Bit-Blasting procedures")

  => Difficult for a lot of operations (e.g. division)
- States are limited in size
- We can use existing, fast SAT solvers for bounded model checking (SATO, MiniSat, z3)

[1]Biere, et al. 1999. Symbolic model checking without BDDs

SMT based encoding[2]

```
3:    ...
4:   a := a + 1
5:    ...
```

$x = (x_0, x_1, x_2)$ and $y = (y_0, y_1, y_2)$

where x_i, y_i are integers, reals, bitvectors, arrays, …

$T(x, y)$ true iff. there is a transition from x to y

$$T(x, y) = (x_0 = 4) \wedge (y_0 = 5) \wedge (y_1 = x_1 + 1) \wedge (y_2 = x_2)$$

[2]Cordeiro, et al. SMT-based bounded model checking for embedded ANSI-C software

SMT based encoding[2]

- Also NP-hard and in many cases fast to solve in practice
- Simplifies the formula a lot, as many data structures are directly representable in SMT solvers like z3
  => Smaller encoding size
- Infinite types like true integers can be encoded
- SMT solvers can reach similar speeds to SAT solvers
  (internally Bit-Blasting might be used)

[2]Cordeiro, et al. SMT-based bounded model checking for embedded ANSI-C software

How to use these encodings: Bounded Model Checking

Unfolding the transition relation:

$$[\![M]\!]_k := I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1})$$

- Only paths of a certain length are considered
  => We can only provide counter-examples
- By iteratively increasing k, we can find the smallest counter-example
- As transition systems are finite, there is an upper bound for k, but:
- It’s either too hard to compute or really large

Demonstration