Introduction to Software Verification

Vincent Penelle

<vpenelle@u-bordeaux.fr>

LaBRI, Université de Bordeaux

September 23, 2019

— *A simple mode of programs* —

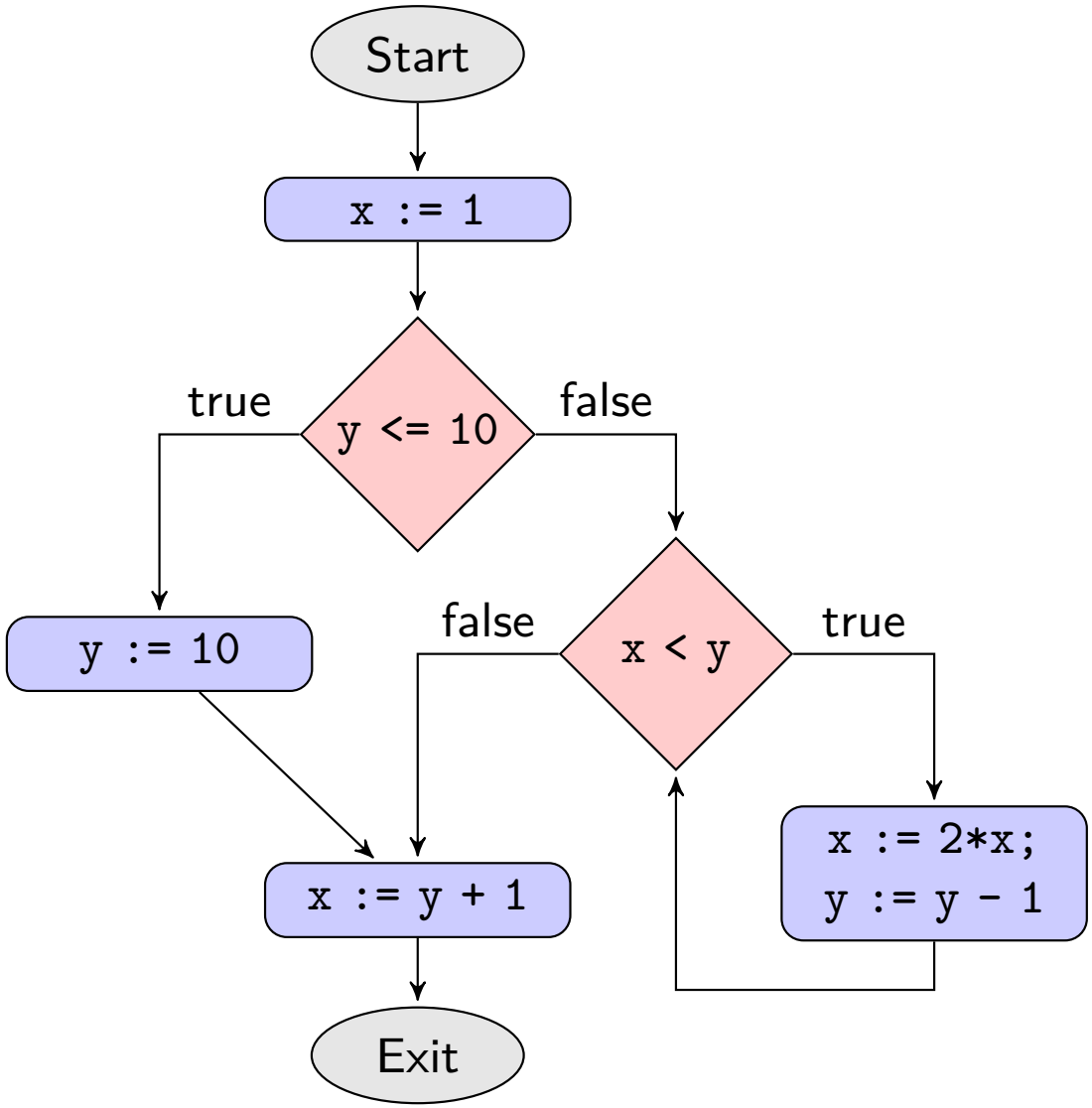
A slight restriction for this course

Today, programs manipulate a variety of data structures that have different size and properties, plus calls to the environment (system). Modelling programs taking that diversity into account and verifying them requires a lot of care, which is a bit tedious, and not necessary to grasp the concept of verification. Moreover, in the end, everything a computer really manipulates are bitvectors (bit arrays), which can be seen as encoding integers.
Therefore, for the purpose of this course, we restrict ourselves to a very simple programming language, C-like, and containing only:

- Int as a type,
- Affectation instructions ($x := 1$),
- Test expression (simple tests, e.g., $x = y * 2$, but not $(x \leq y)\&\&(y + z \leq x)$)
- If then else instructions,
- While instructions.

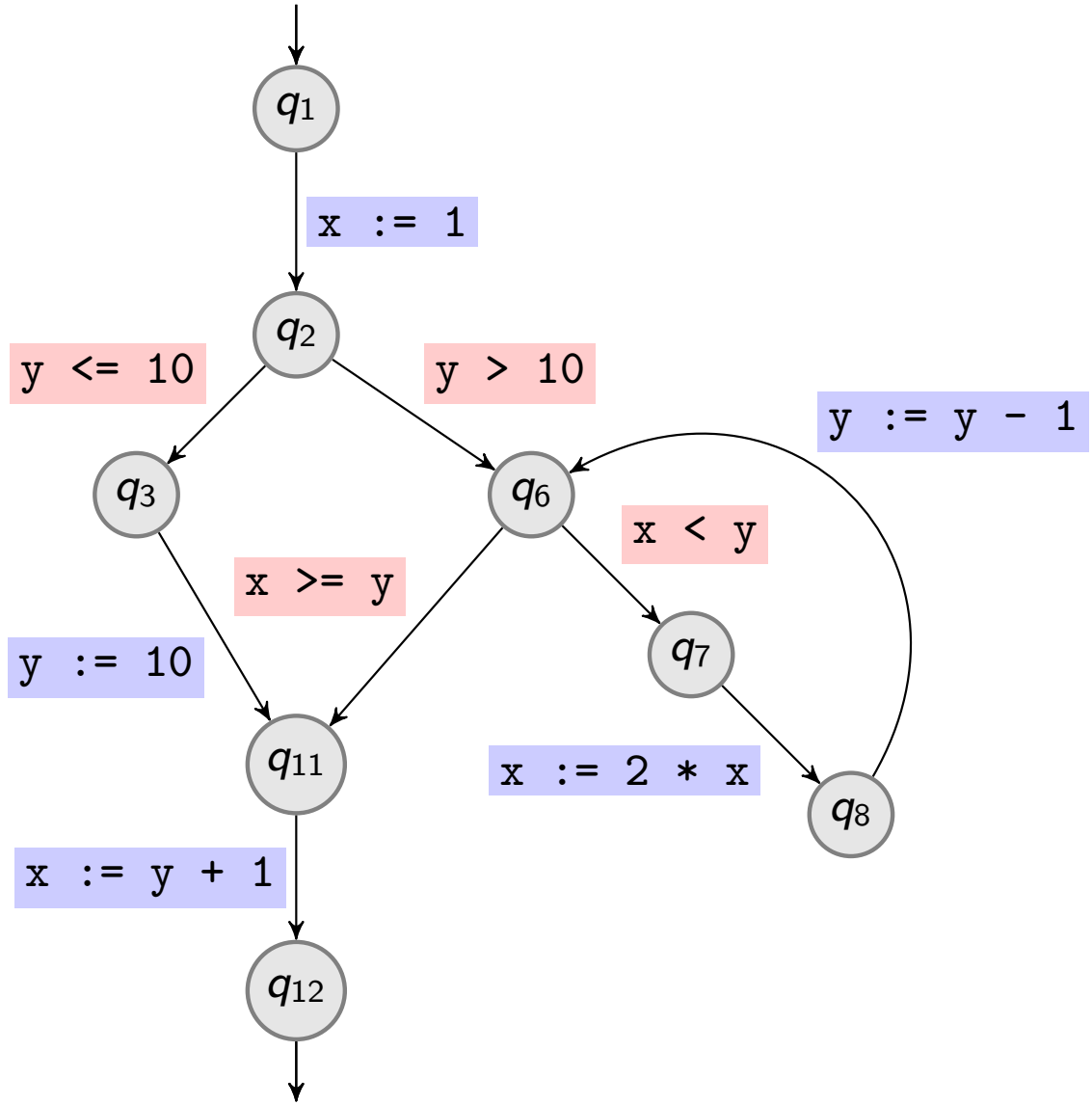
Control Flow Graph

```
x = 1;
if (y <= 10) {
   y = 10;
}
else {
   while (x < y) {
      x = 2 * x;
      y = y - 1;
   }
}
x = y + 1;
```



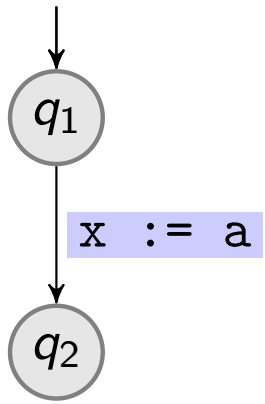
Control Flow Automaton

```
x = 1;
if (y <= 10) {
   y = 10;
}
else {
   while (x < y) {
      x = 2 * x;
      y = y - 1;
   }
}
x = y + 1;
```



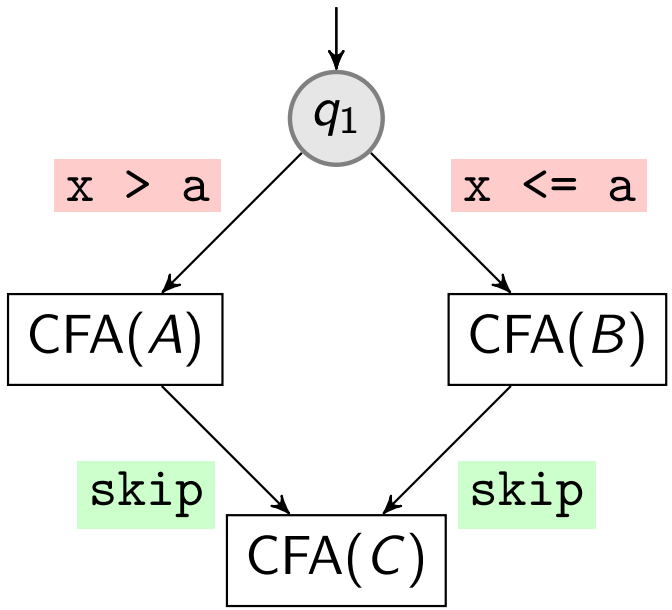
Assignment

```
x = a;
```



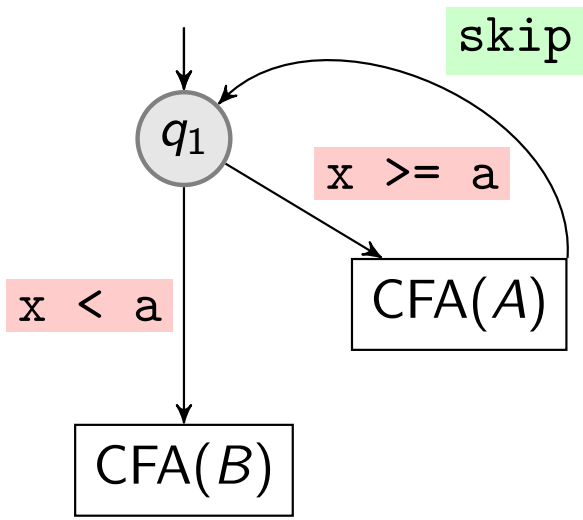
If then else

```
if (x > a){
        A
        }
        else{
        B}
C;
```



While

```
while(x >= a){
        A;
}
B;
```



Semantic of control-flow automata

- Present the notions of transition system (configuration, step, run).
- Present the semantic of different instructions.

Exercise

Give simple programs to translate.
Then onwards to the programming of the tool.

From CFA to SMT

Do it here? In exercise? Directly on the tool?