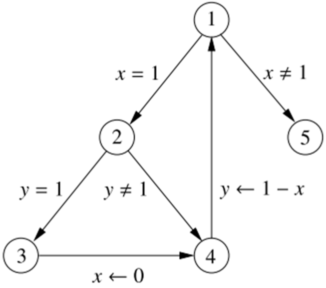
Symbolic exploration

- A technique to palliate the state-explosion problem
- Configurations can be encoded as words.
- The set of reachable configurations of a program can be encoded as a language.
- We use automata to compactly store the set of reachable configurations.

Flowgraphs

```
while x = 1 do
    if y = 1 then
        x ← 0
    y ← 1 − x
end
```



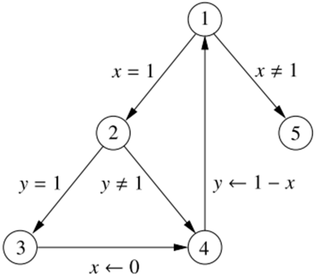
Step relations

- Let l, l' be two control points of a flowgraph.
- The step relation $S_{l,l'}$ contains all pairs

$$([l, x_0, y_0], [l', x'_0, y'_0])$$

of configurations such that :

if at point l the current values of x, y are x_0, y_0,
then the program can take a step,
after which the new control point is l', and the new values of x, y are x'_0, y'_0.



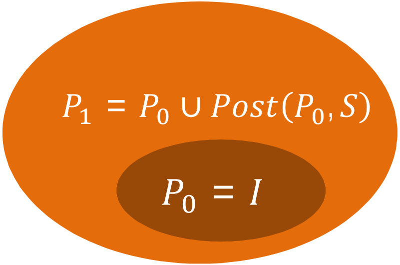
$$S_{4,1} = \{\ (\ [4, x_0, y_0], [1, x_0, 1 - x_0]\ )\ \mid\ x_0, y_0 \in \{0,1\}\ \}$$

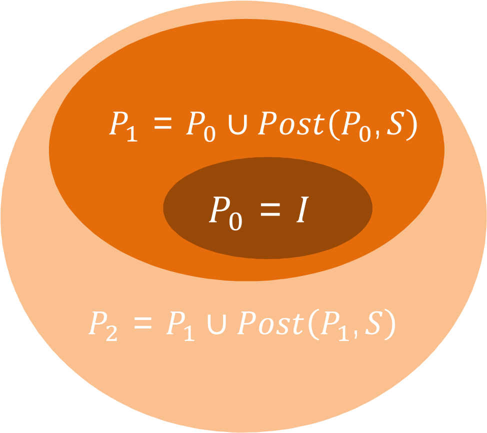
- The global step relation S is the union of the step relations $S_{l,l'}$ for all pairs l, l' of control points.

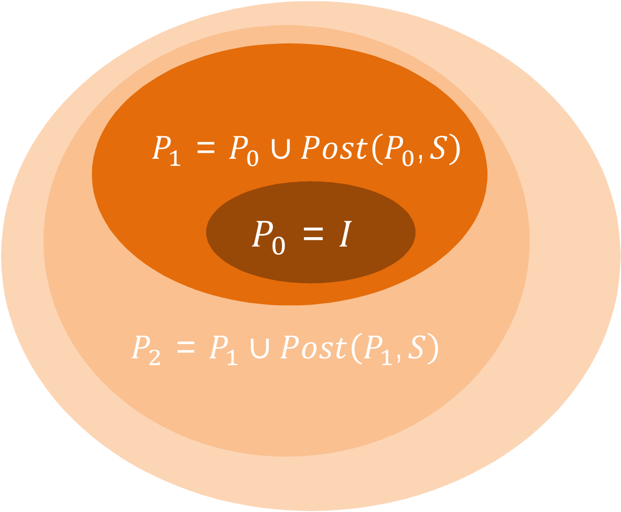
Computing reachable configurations

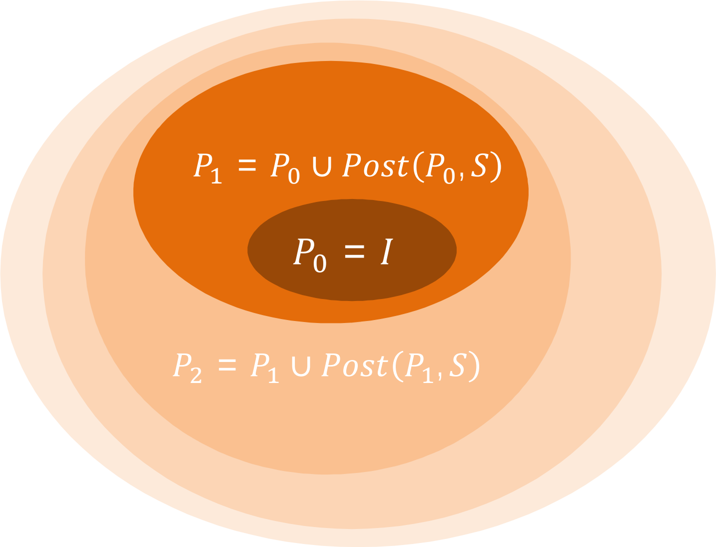
- Start with the set of initial configurations.
- Iteratively: add the set of successors of the current set of configurations until a fixed point is reached.

$$P_0 = I$$


$P_1 = P_0 \cup Post(P_0, S)$
$P_0 = I$


$P_1 = P_0 \cup Post(P_0, S)$
$P_0 = I$
$P_2 = P_1 \cup Post(P_1, S)$


$P_1 = P_0 \cup Post(P_0, S)$
$P_0 = I$
$P_2 = P_1 \cup Post(P_1, S)$

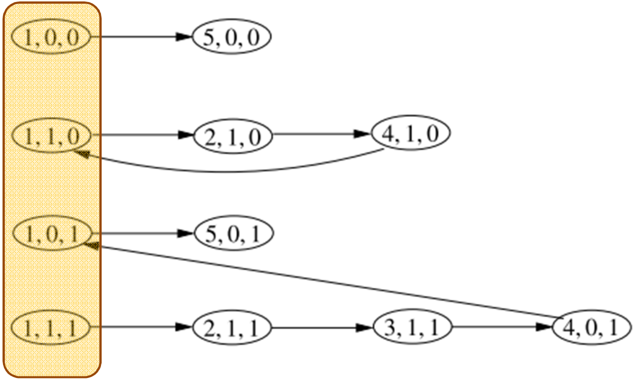

$P_1 = P_0 \cup Post(P_0, S)$
$P_0 = I$
$P_2 = P_1 \cup Post(P_1, S)$

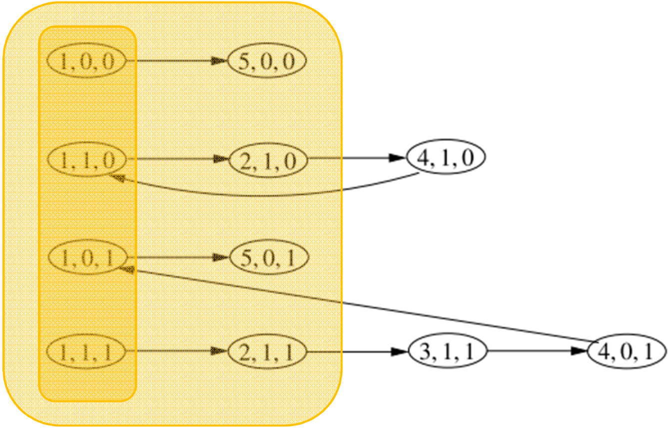
Reach(I, R)

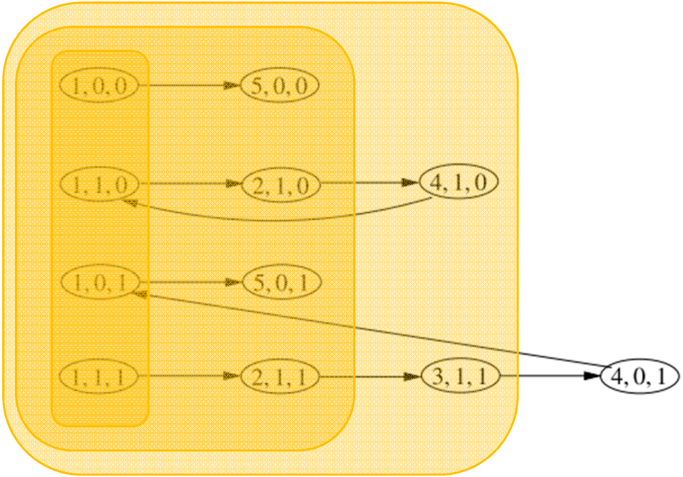
Input: set I of initial configurations; relation R

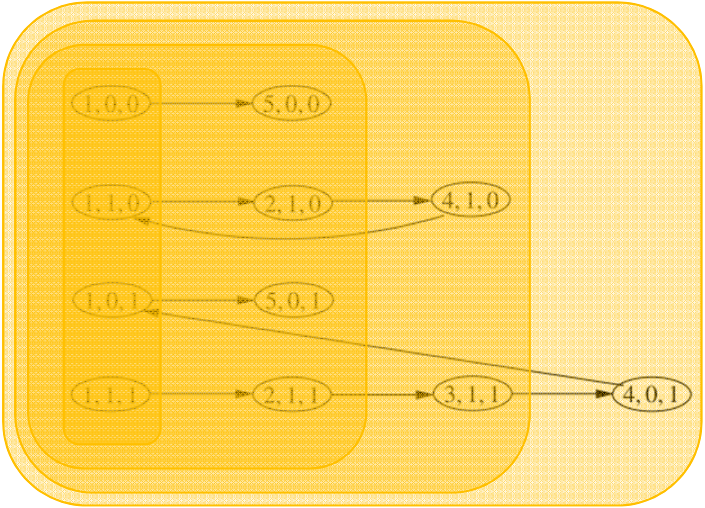
Output: set of configurations reachable form I

$OldP \leftarrow \emptyset$; $P \leftarrow I$
while $P \neq OldP$ **do**
    $OldP \leftarrow P$
    $P \leftarrow$ **Union**$(P,$ **Post**$(P, S))$
return P


1, 0, 0
5, 0, 0
1, 1, 0
2, 1, 0
4, 1, 0
1, 0, 1
5, 0, 1
1, 1, 1
2, 1, 1
3, 1, 1
4, 0, 1

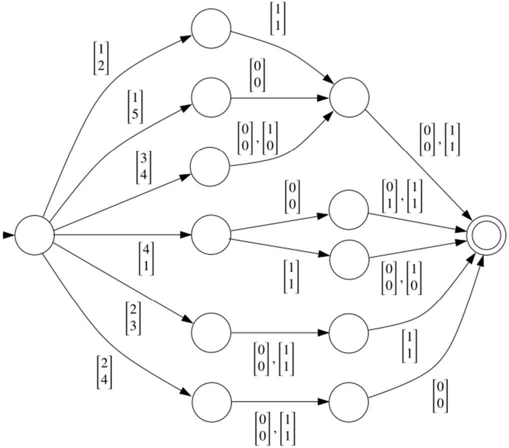

1, 0, 0
5, 0, 0
1, 1, 0
2, 1, 0
4, 1, 0
1, 0, 1
5, 0, 1
1, 1, 1
2, 1, 1
3, 1, 1
4, 0, 1


1, 0, 0
5, 0, 0
1, 1, 0
2, 1, 0
4, 1, 0
1, 0, 1
5, 0, 1
1, 1, 1
2, 1, 1
3, 1, 1
4, 0, 1


1, 0, 0
5, 0, 0
1, 1, 0
2, 1, 0
4, 1, 0
1, 0, 1
5, 0, 1
1, 1, 1
2, 1, 1
3, 1, 1
4, 0, 1

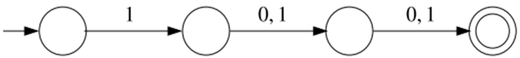
Example: Transducer for the global step relation

```
while x = 1 do
    if y = 1 then
        x ← 0
    y ← 1 − x
end
```

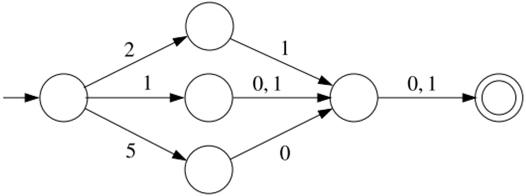


Example: DFAs generated by Reach

- Initial configurations

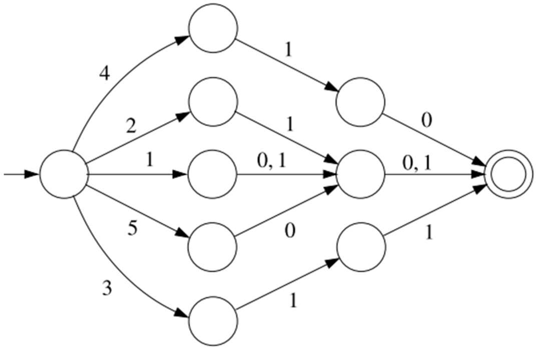


- Configurations reachable in at most 1 step



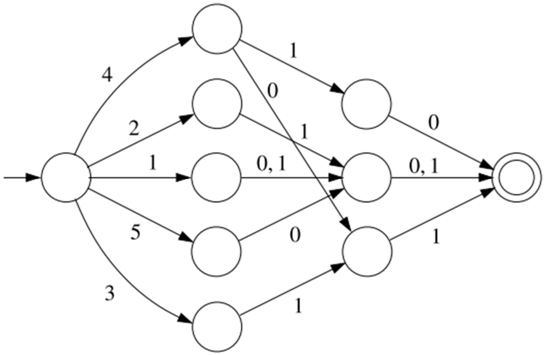
Example: DFAs generated by Reach

- Configurations reachable in at most 2 steps



Example: DFAs generated by Reach

- Configurations reachable in at most 3 steps



Variable orders

- Consider the set Y of tuples $[x_1, \ldots, x_{2k}]$ of booleans such that

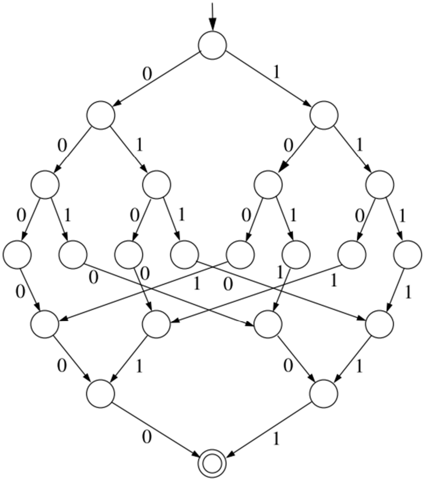
$$x_1 = x_{k+1}, x_2 = x_{k+2}, \ldots, x_k = x_{2k}$$

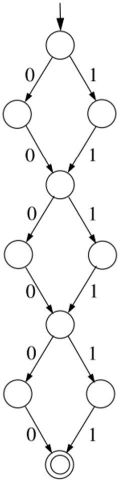
- A tuple $[x_1, \ldots, x_{2k}]$ can be encoded by the word $x_1 x_2 \ldots x_{2k-1} x_{2k}$ but also by the word $x_1 x_{k+1} \ldots x_k x_{2k}$.
- For $k = 3$, the encodings of Y are then, respectively

$$\{000000, 001001, 010010, 011011, 100100, 101101, 110110, 111111\}$$

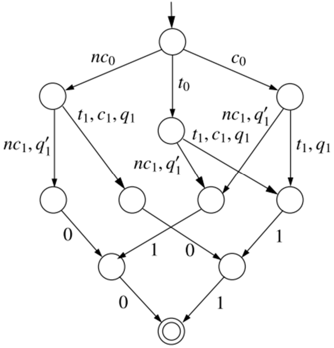
$$\{000000, 000011, 001100, 001111, 110000, 110011, 111100, 111111\}$$

- The minimal DFAs for these languages have very different sizes!

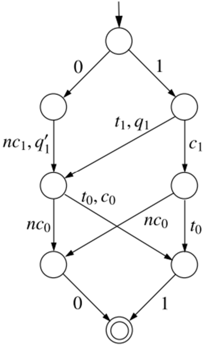

0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1


0
1
0
1
0
1
0
1
0
1
0
1

Another example: Lamport‘s algorithm

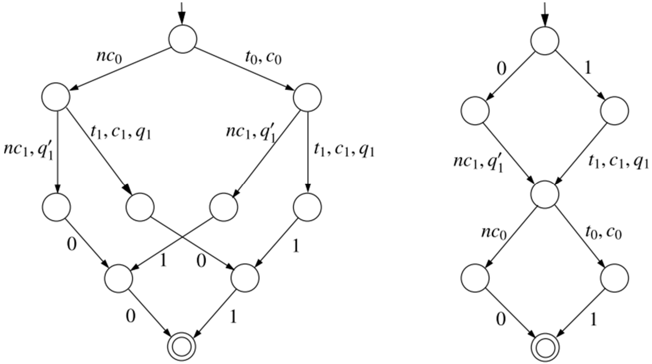


$\langle v_0, v_1, s_0, s_1 \rangle$ encoded by $s_0 s_1 v_0 v_1$



$\langle v_0, v_1, s_0, s_1 \rangle$ encoded by $v_1 s_1 s_0 v_0$

Larger sets can yield smaller DFAs!



- DFAs after adding the configuration $\langle c_0, c_1, 1{,}1\rangle$ to the set

- When encoding configurations, good variable orders can lead to much smaller automata.
- Unfortunately, the problem of finding an optimal encoding for a language represented by a DFA is NP-complete.