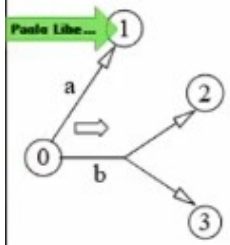
Policies and solutions

example of a non-deterministic action:



two actions can be executed from state `0`:

`a`
always leads to state `1`
deterministic action

`b`
may lead to state `2` or to `3`
nondeterministic action

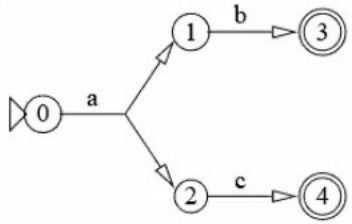
examples

examples of nondeterministic actions

- depend on external entities (es. user)
- actions that may fail
- other?

policy

= what to do in each state

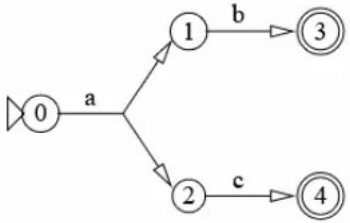


policy:

1. execute `a`
2. if in state `1` execute `b`
   if in state `2` execute `c`

policy, formally

assigment from states to actions



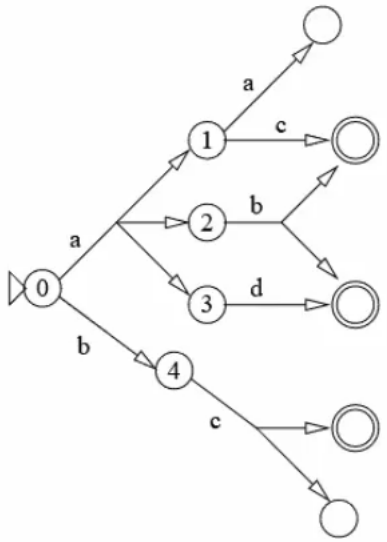
example: `{(0,a), (1,b), (2,c)}`

meaning: `(1,b)` = in state `1` execute action `b`

reaching a goal state

deterministic actions
    solution = path from the initial state to a goal state
nondeterministic actions
    solution = one path? all paths?

good solutions and poor solutions

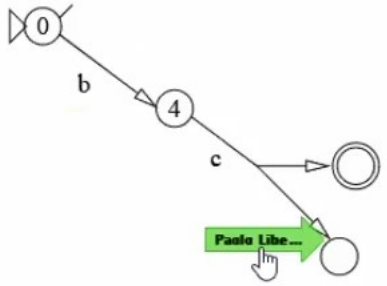


executing `a` looks more complicated:
three possible outcomes to keep into account

with `b`, only one

weak solution

after executing `b`:

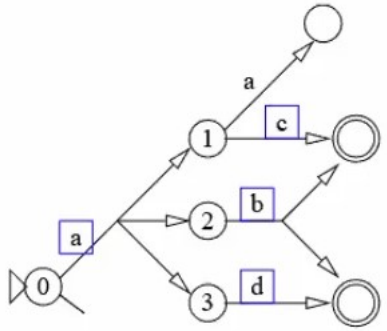


only `c` executable in `4`

may lead to the goal, or not

strong solution

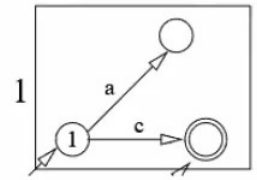
executing `a` instead:



depending on the state, execute `c`, `b` or `d`

in all three cases, goal reached!

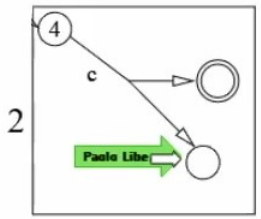
nondeterminism vs. choice



not the same:

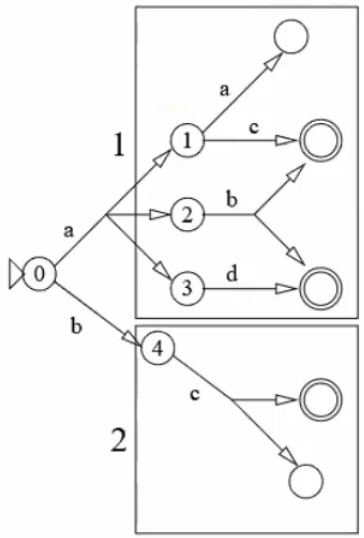
in box 1, we can choose where to go
(with `a` we go to the non-goal state,
with `c` to the goal state)

in box 2, nondeterminism decides where to go
(only applicable action is `c`)



{(0,a),(1,c),(2,b),(3,d)>

good solutions and poor solutions



in the initial state:

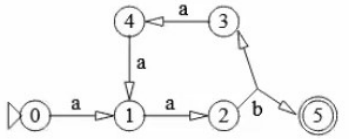
1. first do `a`, then either `c`, `b` or `d`:
   goal is always reached
   strong (acyclic) solution

2. do `b` and `c`:
   goal may be reached
   weak solution

strong solutions are better

why "acyclic"?

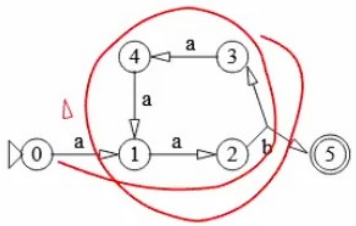
the trouble with cycles



goal always reachable?

never say goal

not sure the goal can be reached



action `b` is non deterministic

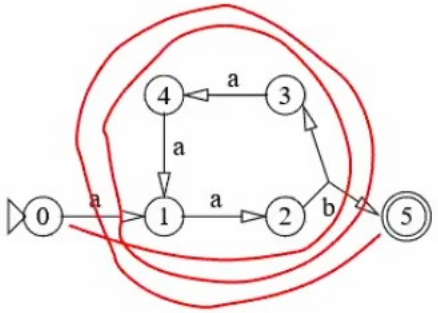
no guarantee that it ever lead to `5`
no guarantee that the goal will ever be reached

but…

never say never

the goal is always reachable

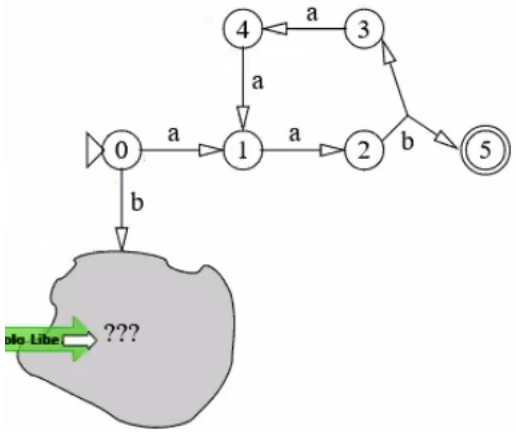
after 1000 loops, maybe the next will be the lucky one

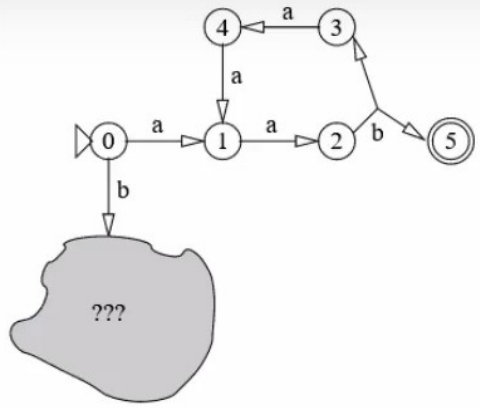


what to do with cycles?

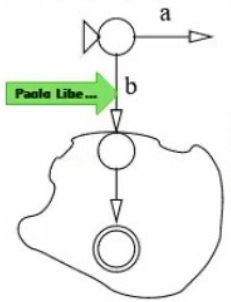
previous example: no choice
enter the loop and hope

what about:

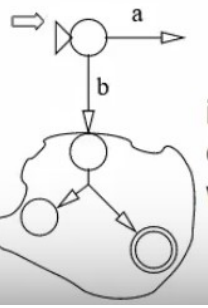




depending on what is inside ???:



in ??? we can always reach the gaol, with no cycles
go there (do `b`)



in ???, only weak solutions
enter the cycle instead (do `a`)
why: in ??? we may reach the goal but also end up where the goal is unreachable

solutions with cycles vs. weak solutions

1. no solution ⇒ bad
2. weak solution ⇒ maybe we'll reach the goal ⇒ better
3. strong solution with cycles ⇒ maybe we'll reach the goal, but if we don't we can keep trying ⇒ still better
4. strong solution without cycles ⇒ we'll reach the goal ⇒ good

strong solutions, strong acyclic solutions

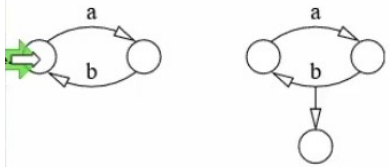
definitions:

strong solution
    goal is always reached, no matter of the nondeterministic choices
strong cyclic solution
    goal might not be reached due to nondeterminism, but is always reachable
weak solution
    goal may be reached or not depending on nondeterminism

strong acyclic solutions ⊂ strong solutions ⊂ weak solutions

word "cyclic" is misleading: a strong cyclic solution may not contain cycles!

bad cycles and good cycles

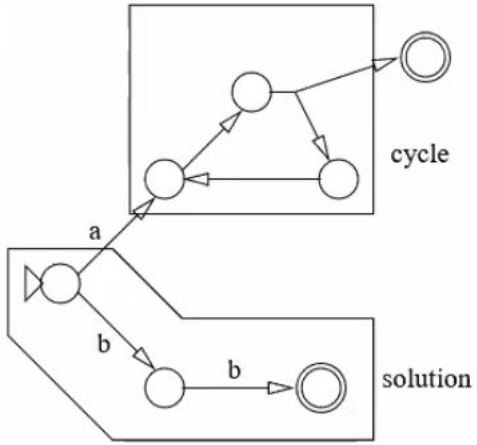


the cycle on the left can never be escaped
a strong cyclic solution cannot include it

the cycle on the right may be escaped
depends on nondeterminism
⇒ a strong cyclic solution can include it

in general: a cycle made only of deterministic actions is never part of a strong cyclic solution
more generally: set of non-goal states with no outgoing arrows

strong (acyclic) solutions and cycles

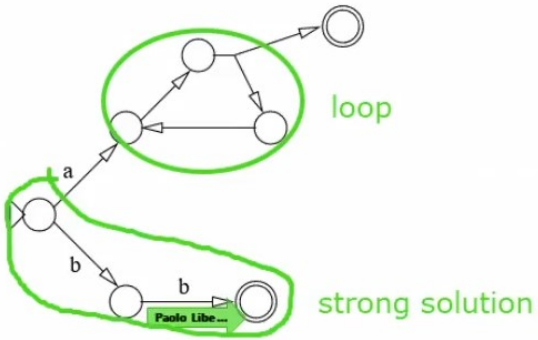


this case: strong (acyclic) solution and cycle separated

always the case?

acyclic solution vs. acyclic problem

a problem may have cycles, but the solution does not include them:

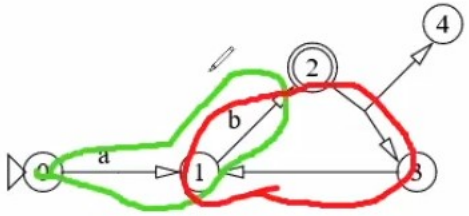


execute `b` and then `b` = solution with no cycle

the problem has cycles
but a strong (acyclic) solution exists

part of a cycle

a cycle may intersect a strong (acyclic) solution:



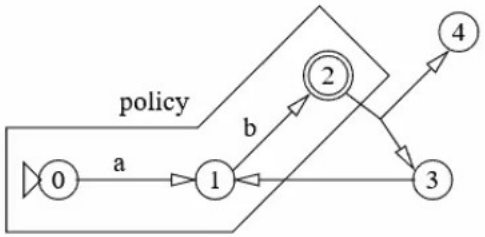
just do `a` and `b`, then stop

policy interseects the cycle
does not include all of it

relevance to a policy

not all states are reached by a policy

in the example:

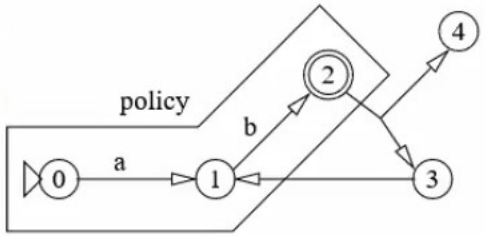


policy reaches `0`, `1` and `2`

stops at `2` (no action there)

does not reach `3` and `4`

states of a policy



policy, formally: `{(0,a), (1,b)}`

tells the action to do in states `1` and `2`

no need to tell what to do in `3`, `4` and `5`:

- `3` is final
  assumption is that we stop at final states
- `4` and `5` are never reached