Progress Control

EECS 20

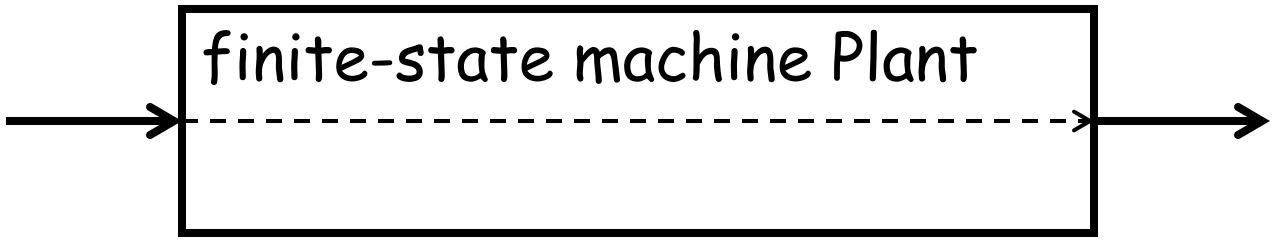
Lecture 37 (April 25, 2001)

Tom Henzinger

Revised by Matteo Zavatteri and Tiziano Villa, Fall 2020

The Safety Control Problem

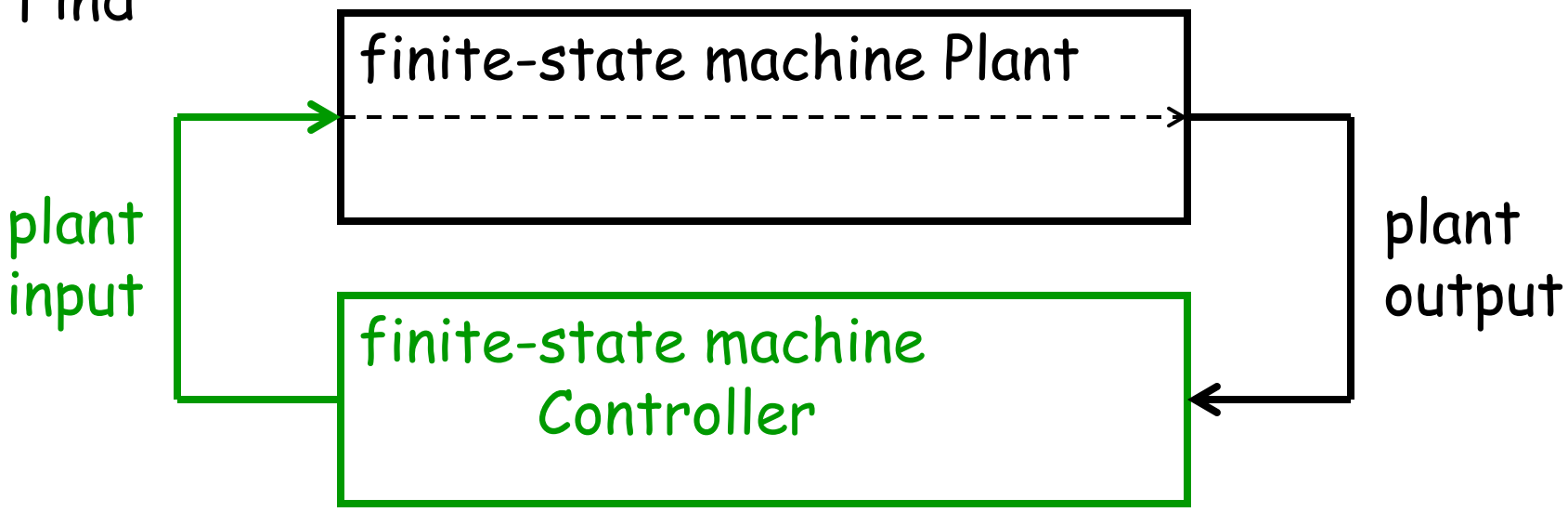
Given

1. 

2. set Error of plant states

The Safety Control Problem

Find



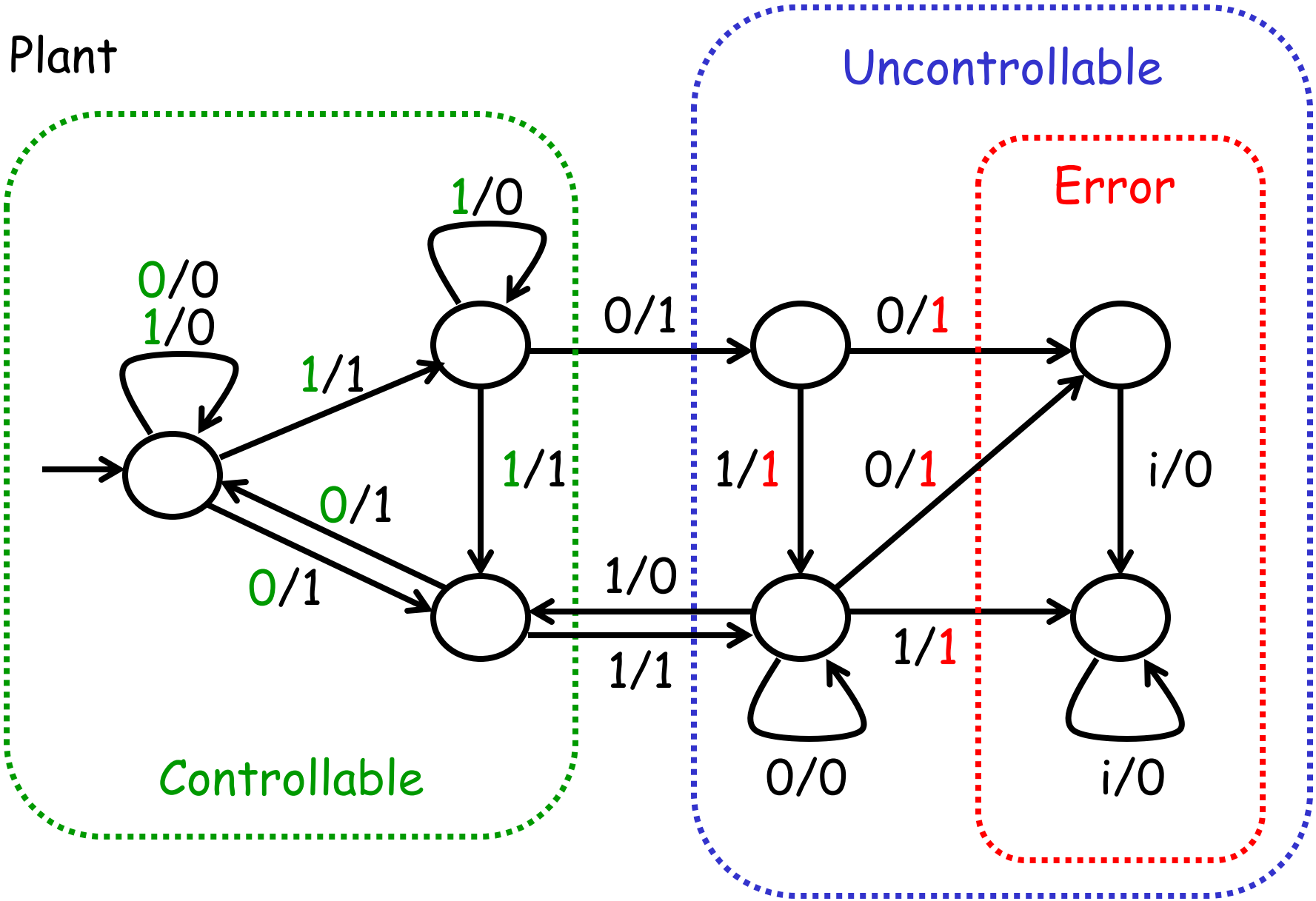
such that the composite system never enters a state in Error

Control is a Game : Plant vs. Controller

Each round consists of two moves:
first Controller chooses plant input,
then Plant chooses plant output

Controllable plant states : controller has a strategy to meet the objective (avoid error states)

Uncontrollable plant states: plant has a strategy to violate the objective (reach an error state)


Plant
Uncontrollable
Error
1/0
0/0
1/0
0/1
0/1
1/1
1/1
0/1
i/0
0/1
0/1
1/0
1/1
1/1
0/0
i/0
Controllable

Controller objective = SAFETY :

stay away from the states in the set Error

Plant objective = PROGRESS :

get to a state in the set Error

The dual control problem:

PROGRESS

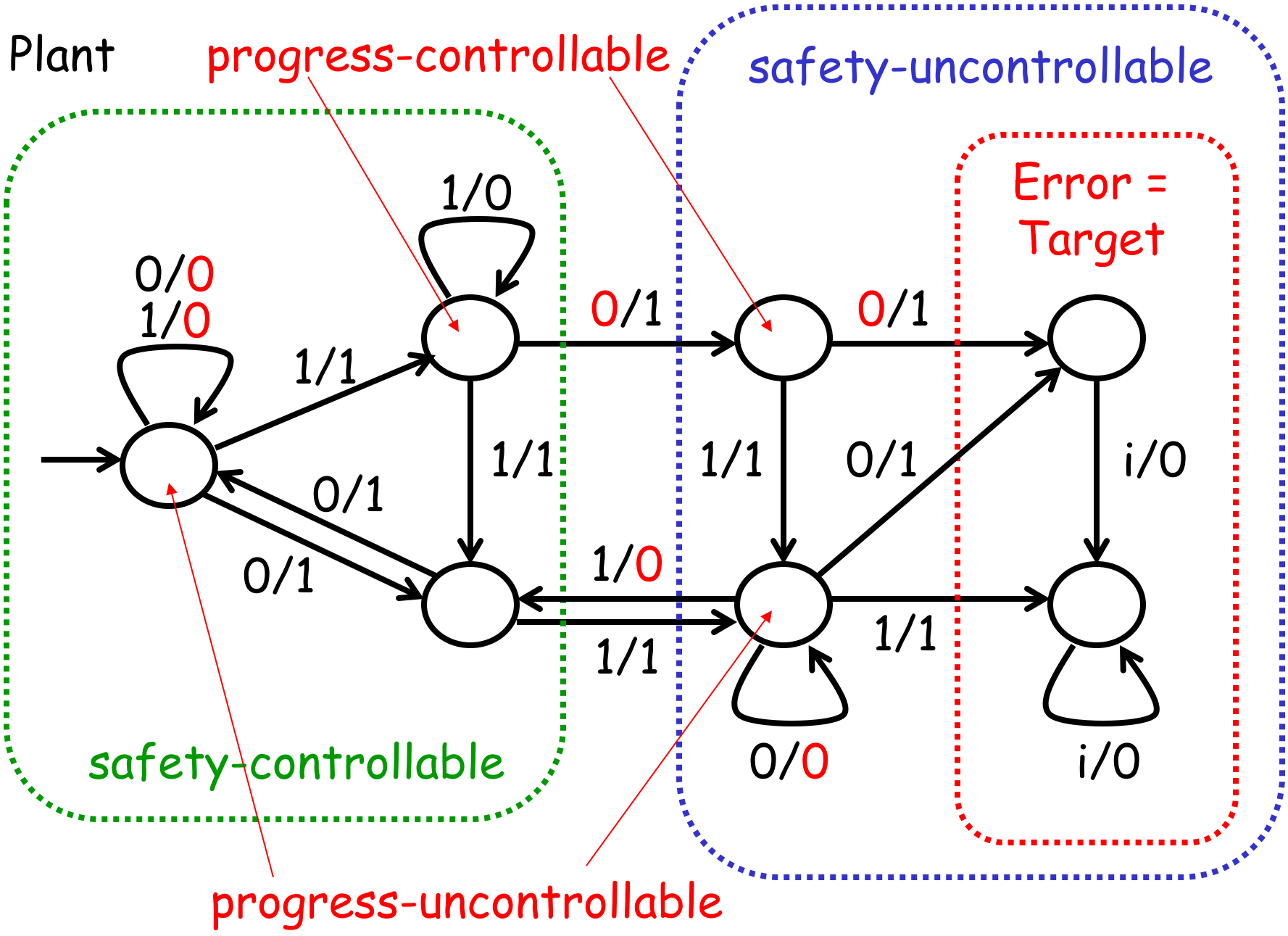
controller attempts to lead the plant into a specified set of states (the "target" states)

Safety Control vs. Progress Control

The roles of Plant and Controller are reversed.

But the progress-controllable states are not the safety-uncontrollable, because the game is not symmetric (the controller always moves first).

Still, the solutions are very similar.


Plant
progress-controllable
safety-uncontrollable
Error =
Target
1/0
0/0
1/0
0/1
0/1
1/1
1/1
1/1
0/1
i/0
0/1
0/1
1/0
1/1
1/1
0/0
i/0
safety-controllable
progress-uncontrollable

Recall Safety Control Step 1:

Compute the safety-uncontrollable states of Plant

1. Every state in Error is safety-uncontrollable.
2. For all states s,

   if for all inputs i
   there exist a safety-uncontrollable state s' and an output o
   such that $(s', o) \in possibleUpdates(s, i)$

   then s is safety-uncontrollable.

Progress Control Step 1:

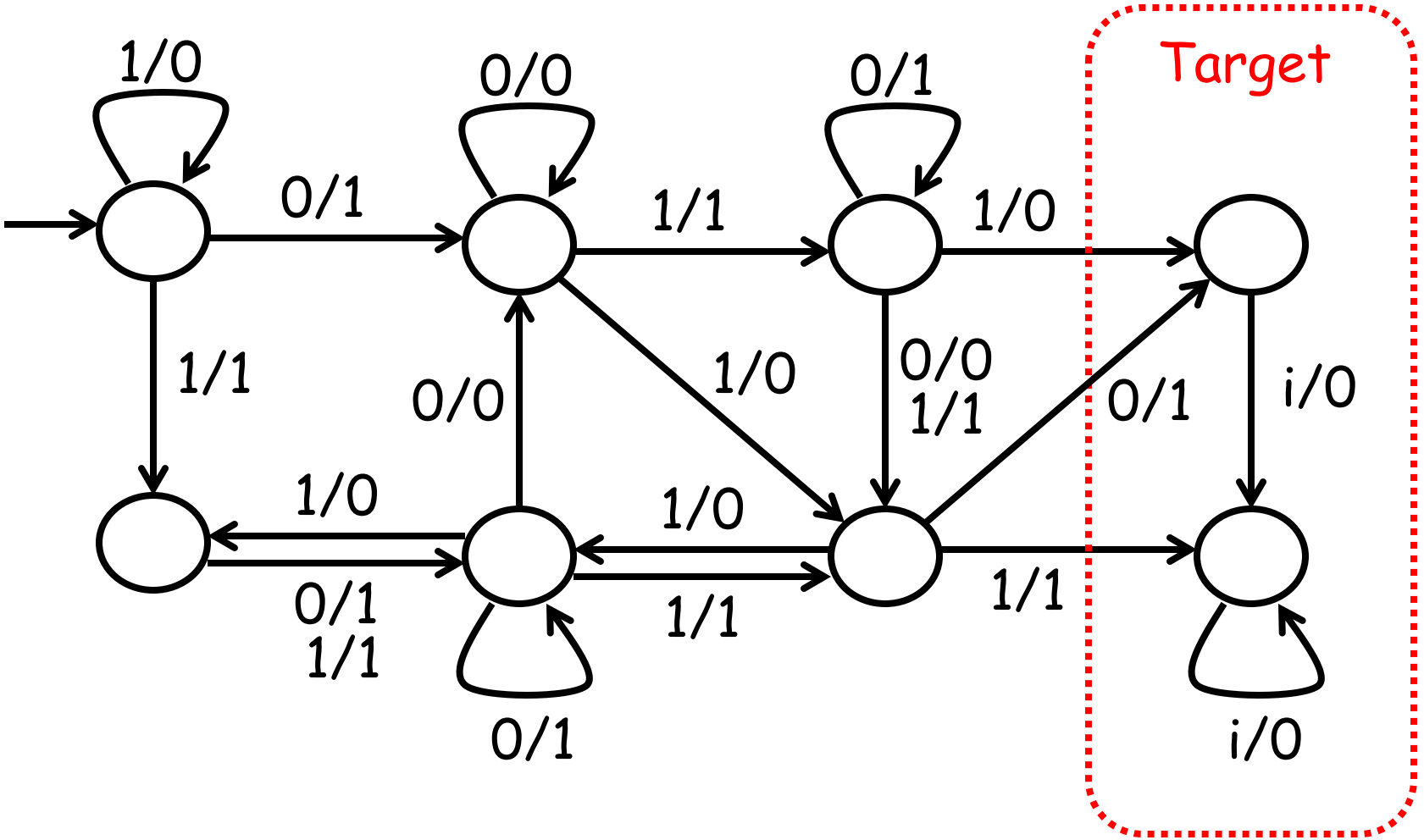
Compute the progress-controllable states of Plant

1. Every state in Target is progress-controllable.
2. For all states s,

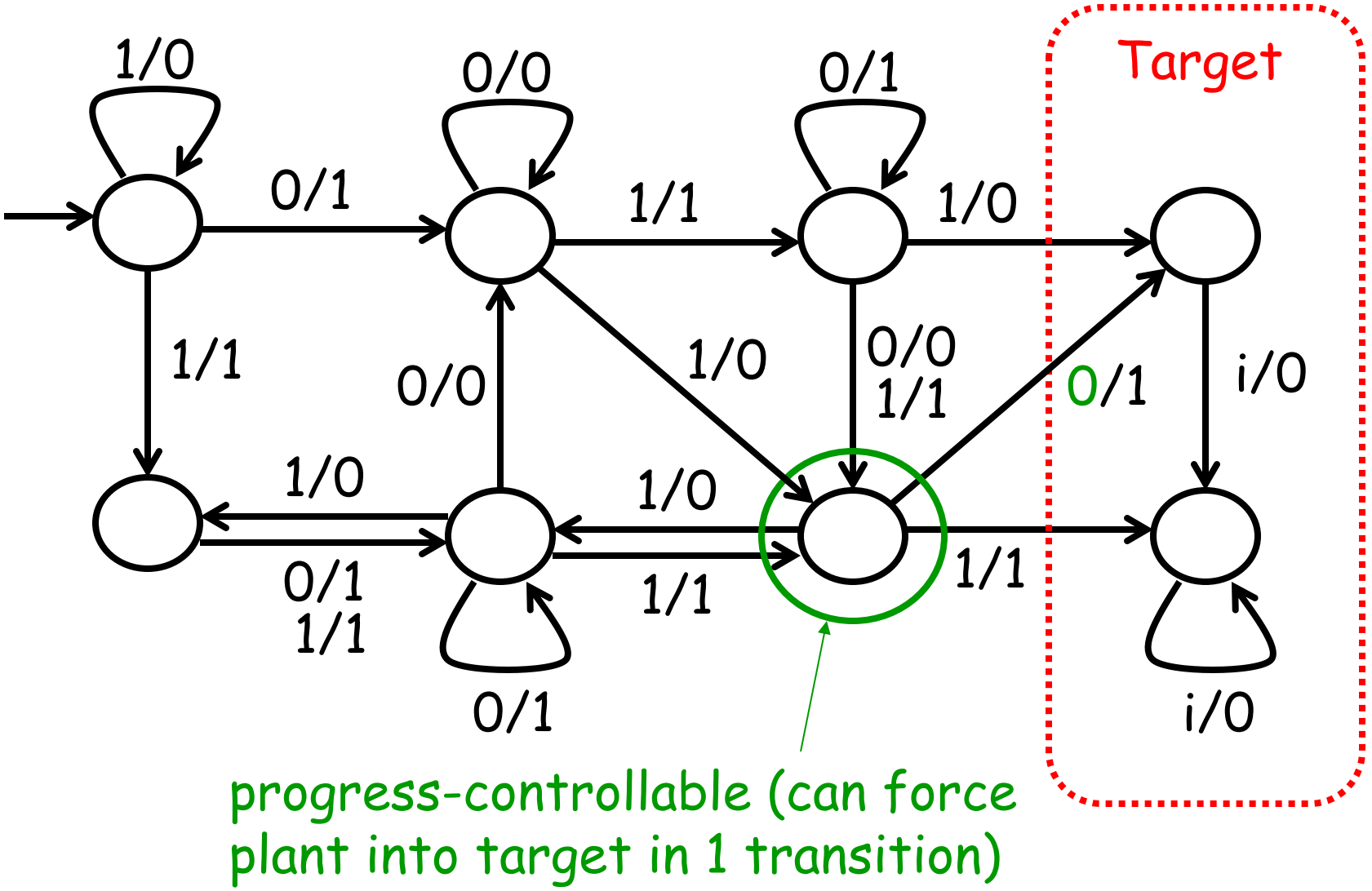
   if there exists an input i
   for all states s' and outputs o
   if $(s',o) \in$ possibleUpdates (s,i)
   then s' is progress-controllable

   then s is progress-controllable.

Plant

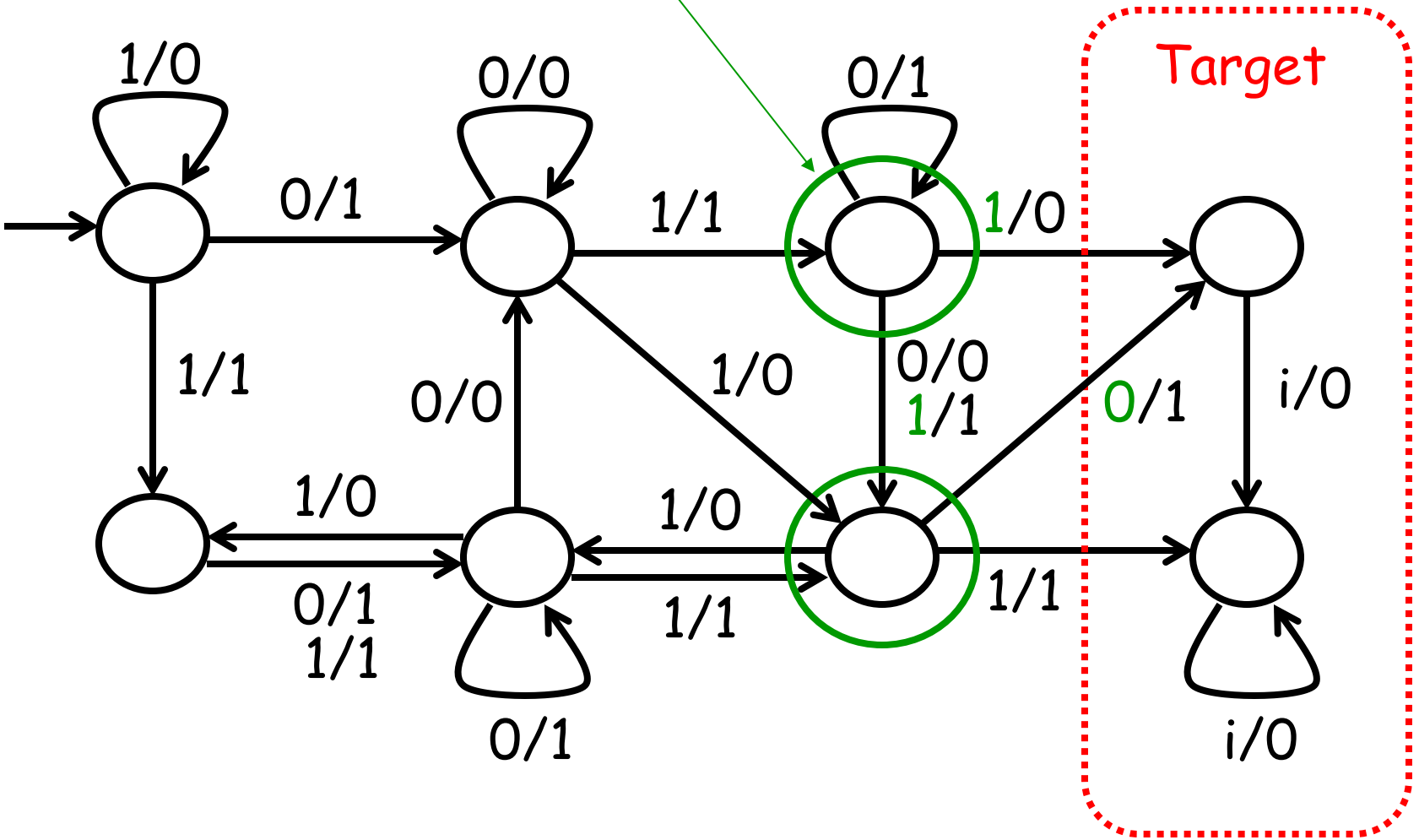


Plant


Target
1/0
0/0
0/1
0/1
1/1
1/0
1/1
0/0
1/0
0/0
1/1
0/1
i/0
1/0
1/0
0/1
1/1
1/1
1/1
0/1
i/0
progress-controllable (can force plant into target in 1 transition)

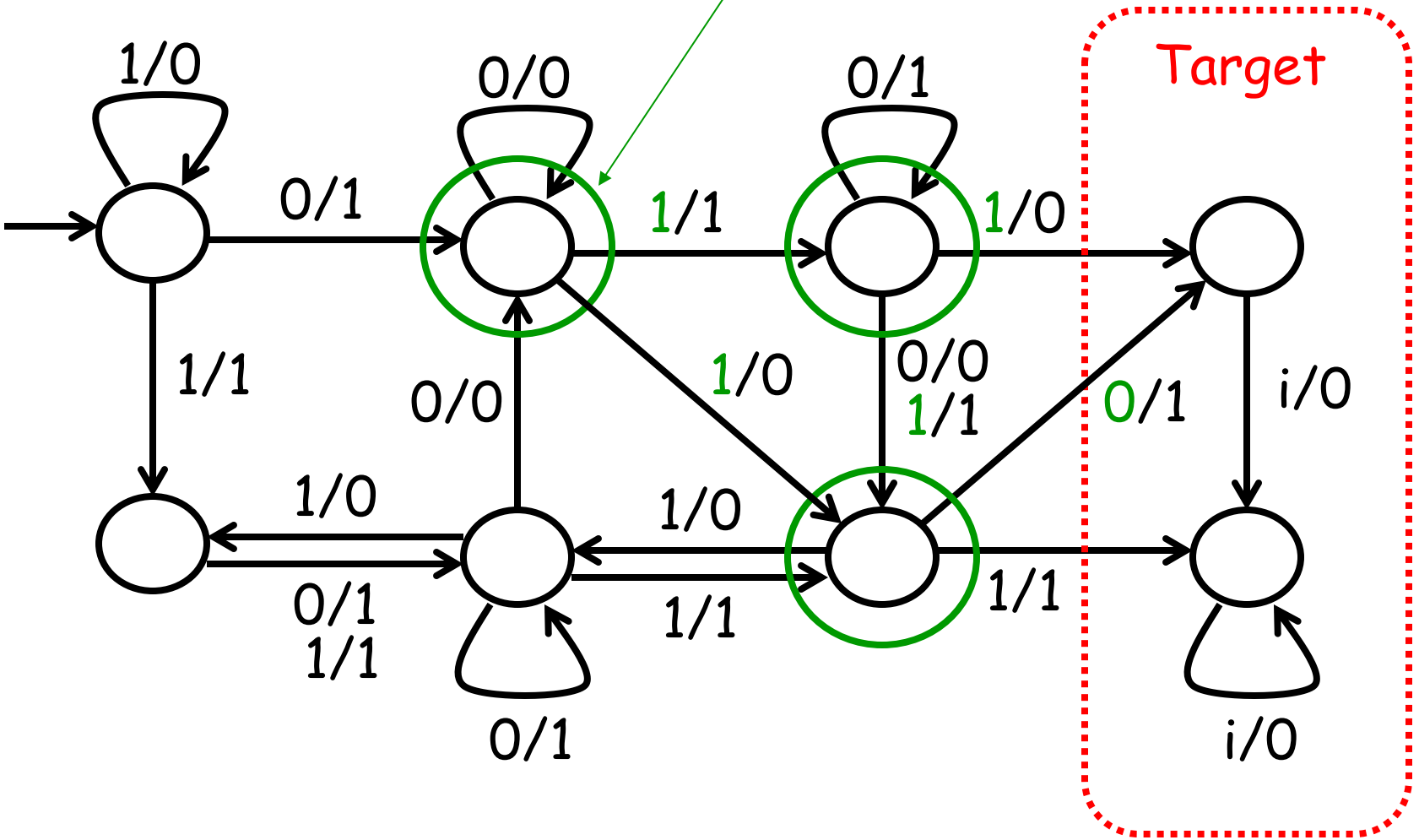
Plant

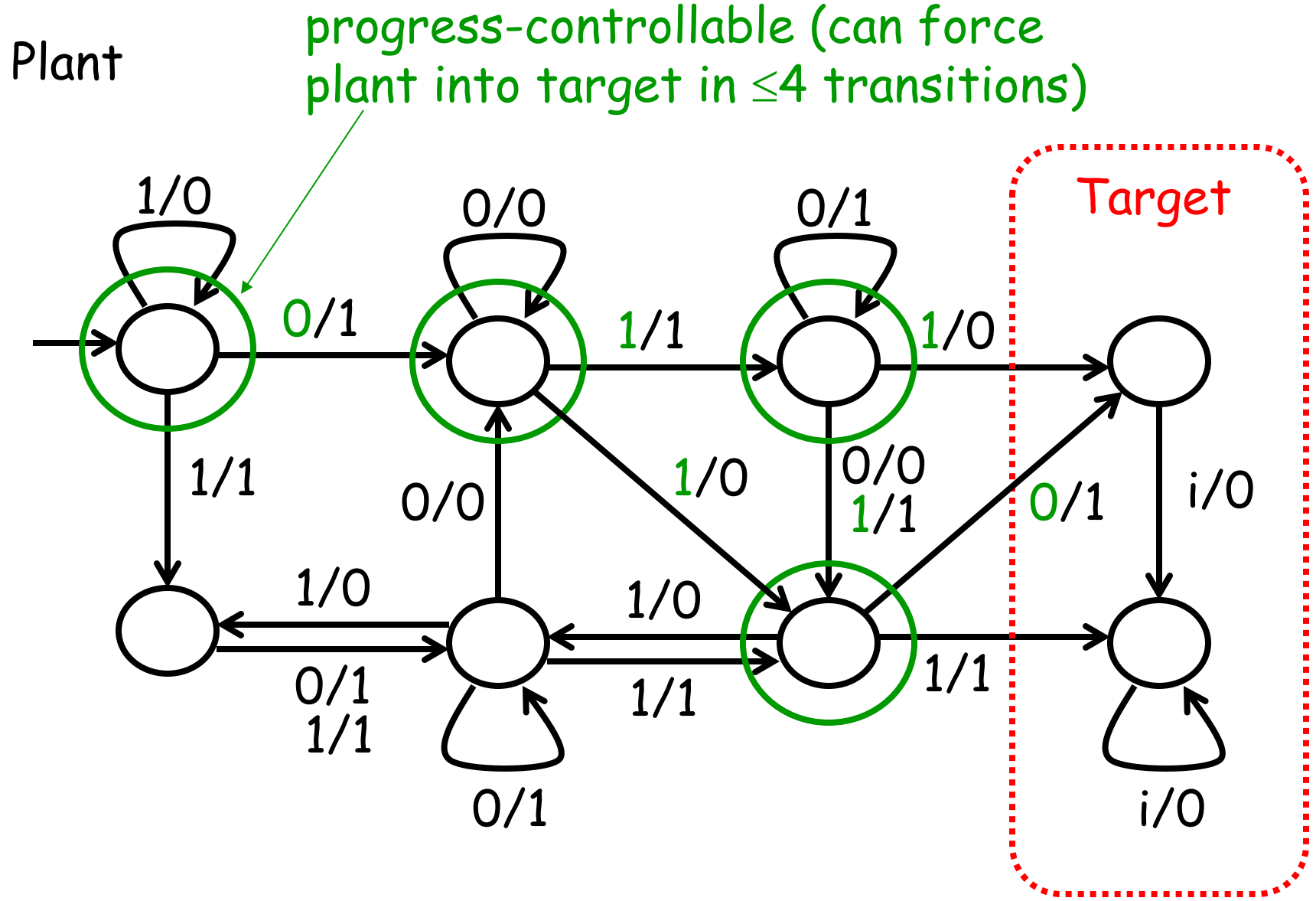
progress-controllable (can force plant into target in ≤2 transitions)



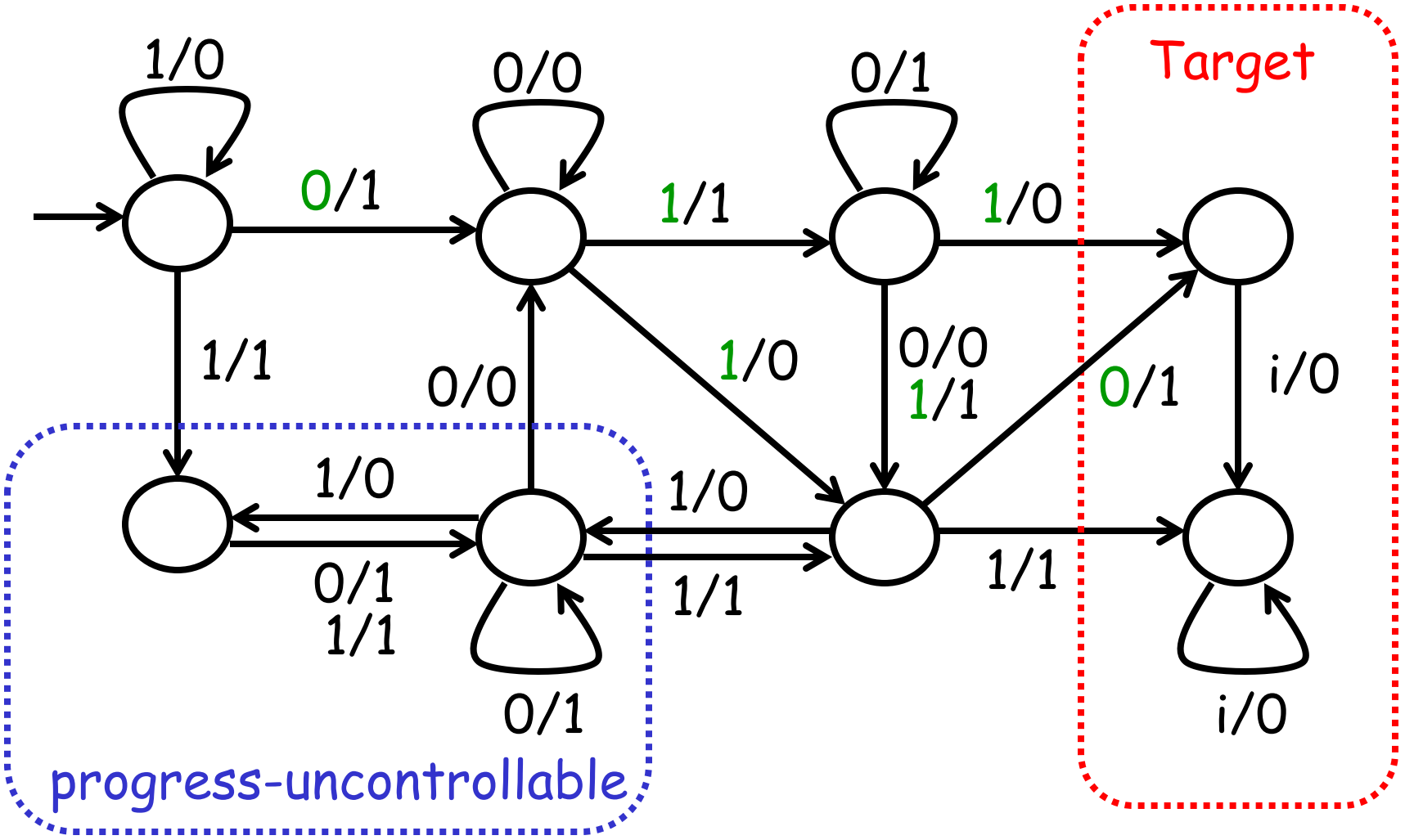
Plant

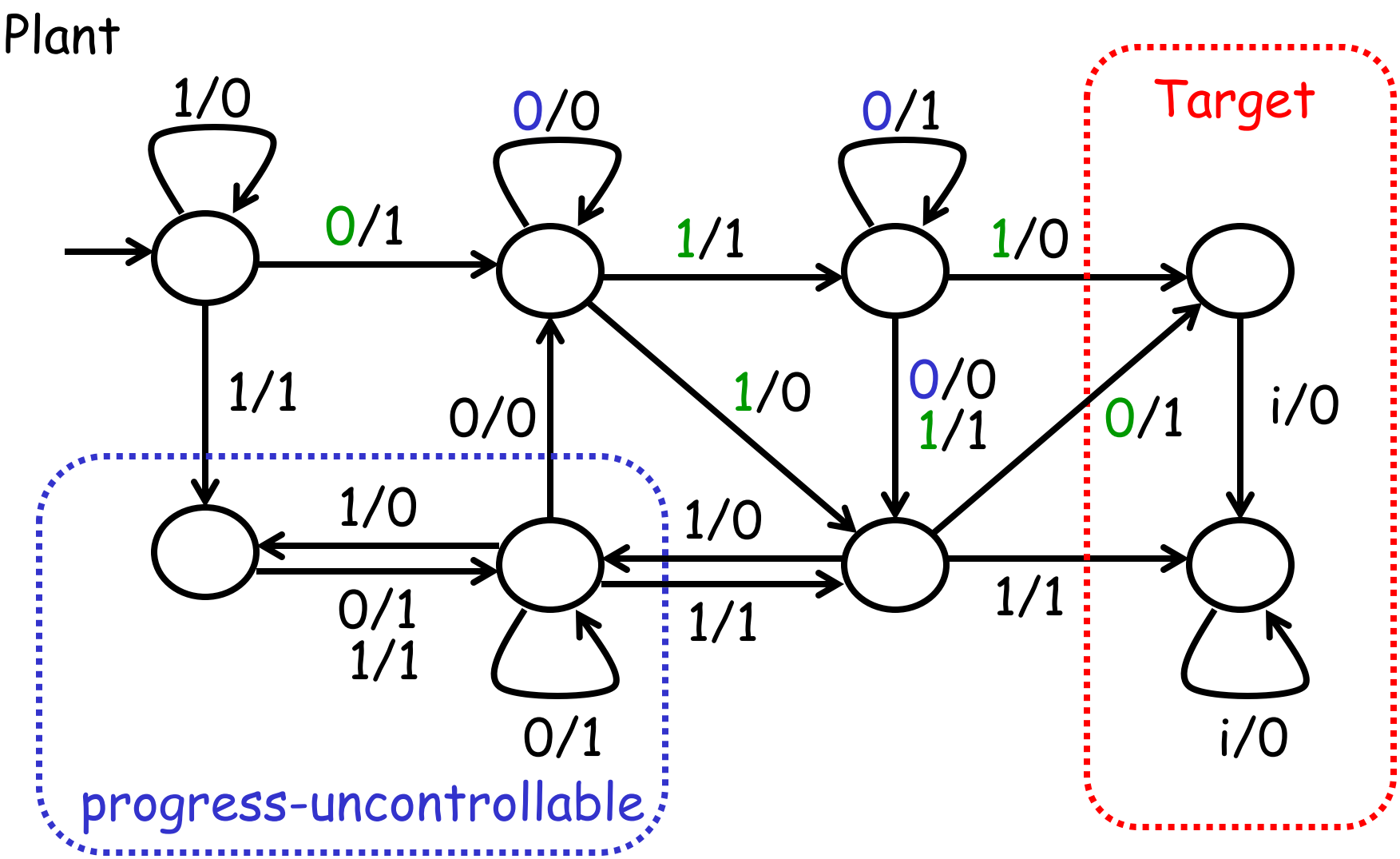
progress-controllable (can force plant into target in ≤3 transitions)




Plant
progress-controllable (can force plant into target in ≤4 transitions)
Target
1/0
0/1
0/0
1/1
0/1
1/0
1/1
0/0
1/1
1/0
0/0
1/1
0/1
i/0
1/0
1/0
0/1
1/1
1/1
1/1
0/1
i/0

Plant


1/0
0/1
0/0
1/1
0/1
1/0
Target
1/1
0/0
1/0
0/0
1/1
0/1
i/0
1/0
1/0
0/1
1/1
1/1
1/1
0/1
i/0
progress-uncontrollable



green: helpful inputs (ensure progress towards target)
blue: safe inputs (keep plant out of uncontrollable states)

Recall Safety Control Step 2:

Track consistent set of safety-controllable plant states

1. A subset $S \subseteq$ Safety-controllable is consistent if there is an input i such that for all states $s \in S$, all states in possibleUpdates (s,i) are safety-controllable.
2. Prune from the state machine whose states are the consistent subsets of Safety-controllable and whose outputs are safe the states without successors.
3. If the result contains possibleInitialStates (of the plant) as a state, then it is the desired Controller. Otherwise, no controller exists.

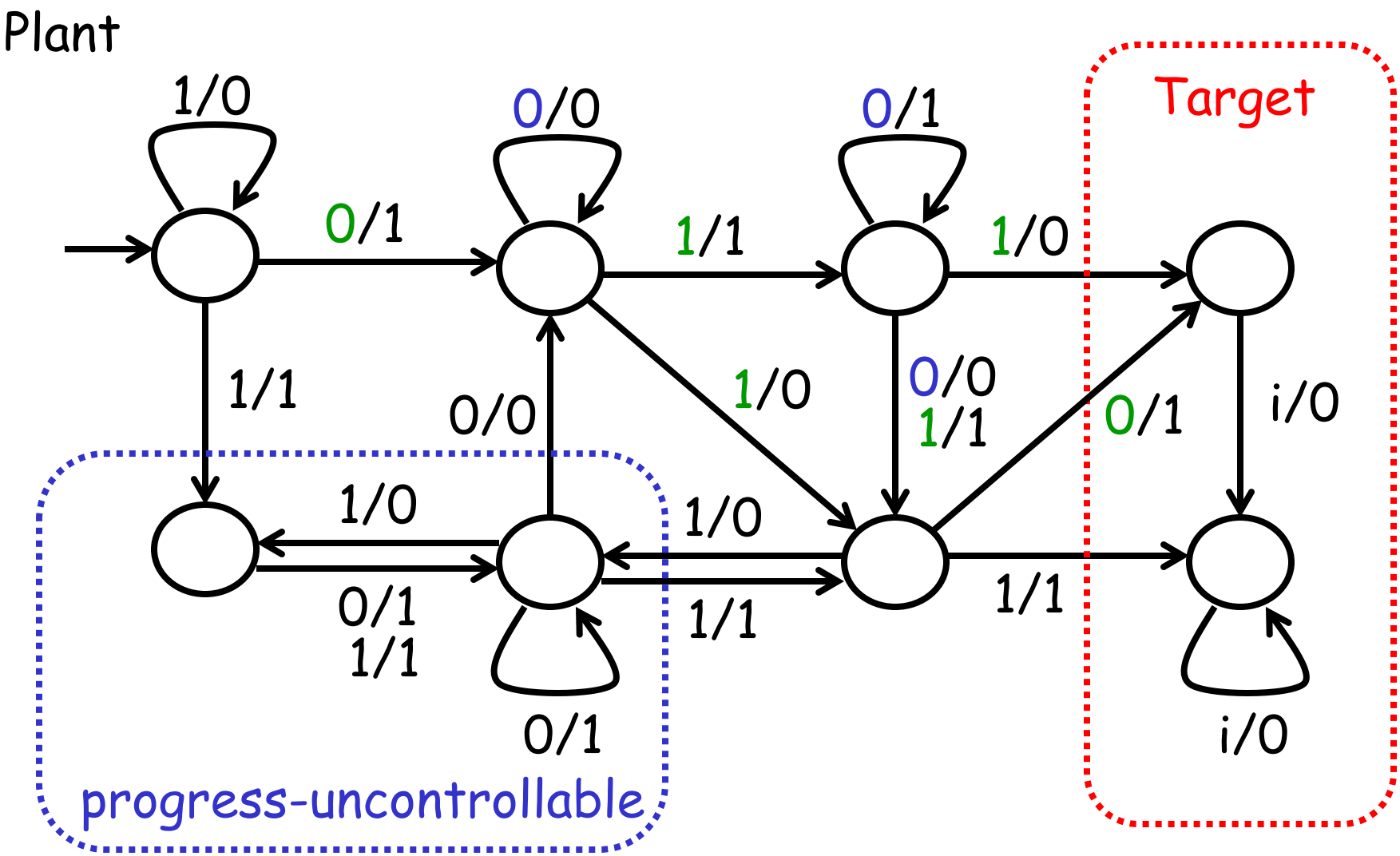
Progress Control Step 2:

Track consistent set of progress-controllable plant states

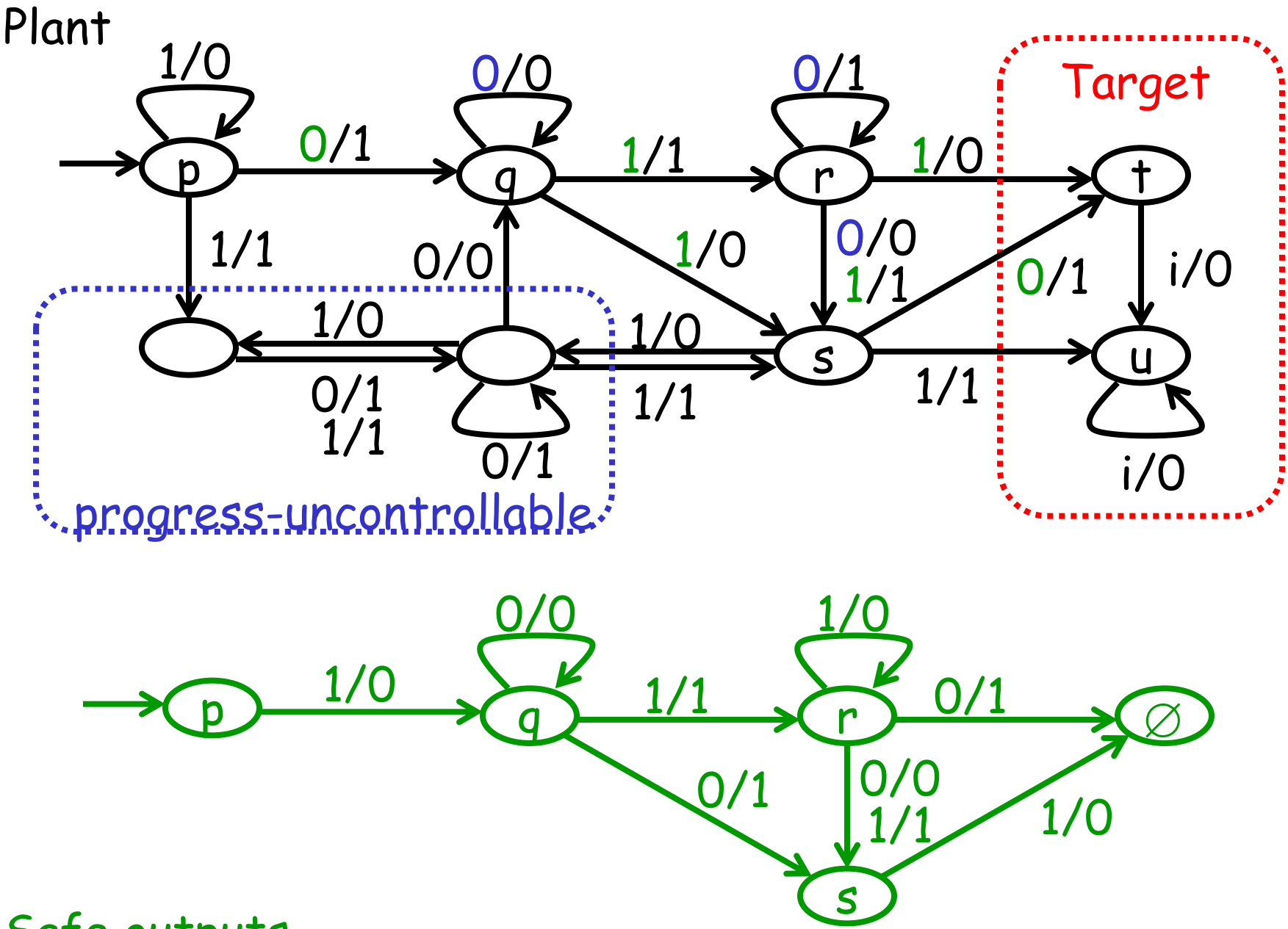
1. A subset $S \subseteq$ Progress-controllable is consistent if there is an input i such that for all states $s \in S$, all states in possibleUpdates (s,i) are progress-controllable.
2. Construct the state machine whose states are the consistent subsets of Progress-controllable without target states (including the empty set Ø), and whose outputs are safe. A safe output is a controller output (and so a plant input) which cannot lead the plant to a progress-uncontrollable state (so safe outputs are blue and green plant inputs).
3. If the result contains possibleInitialStates (of the plant) as a state, and there is an acyclic, output-closed subgraph from possibleInitialStates to Ø, then prune away all states not in the subgraph; this is the desired Controller. Otherwise, no controller exists. A subgraph is output-closed if transitions with safe but not helpful outputs are removed.

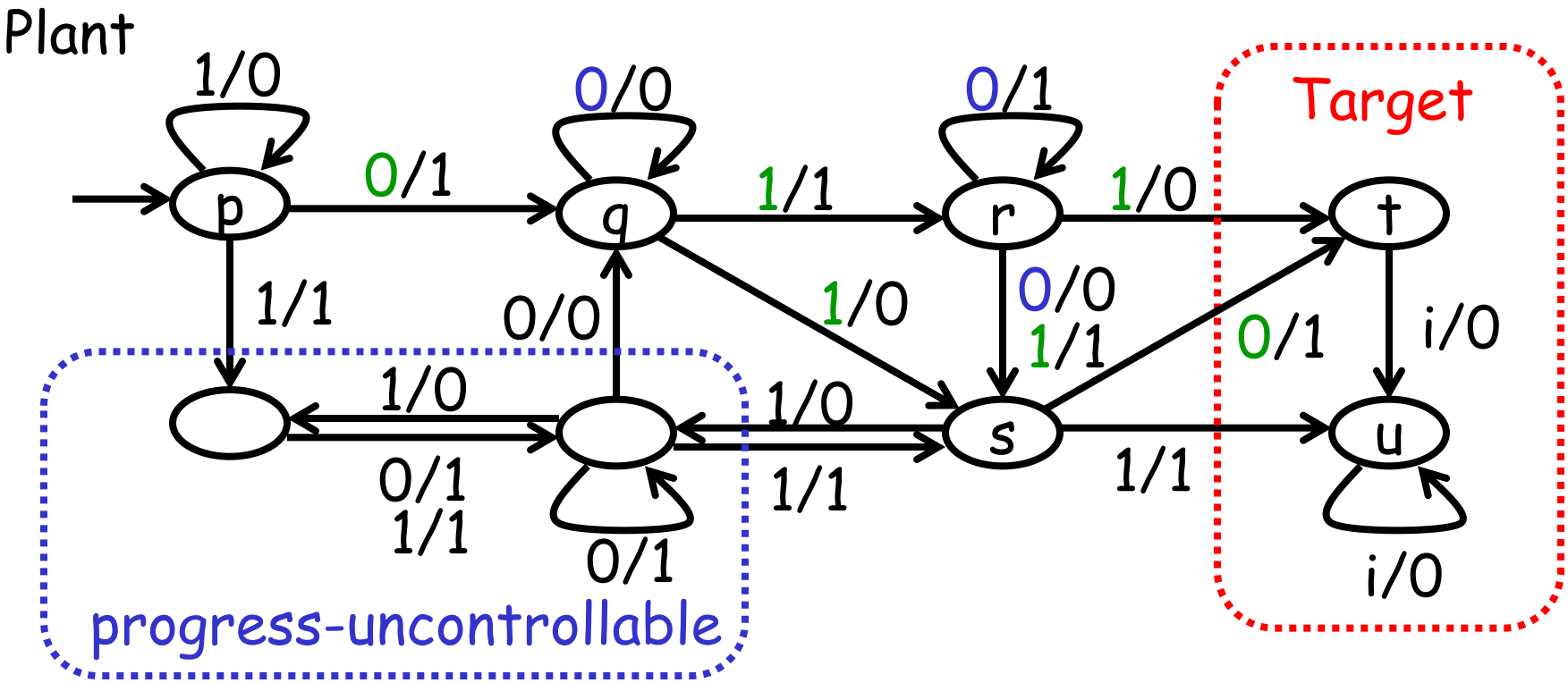
As usual, if the plant is output-deterministic, then we need consider only consistent sets of size 1.

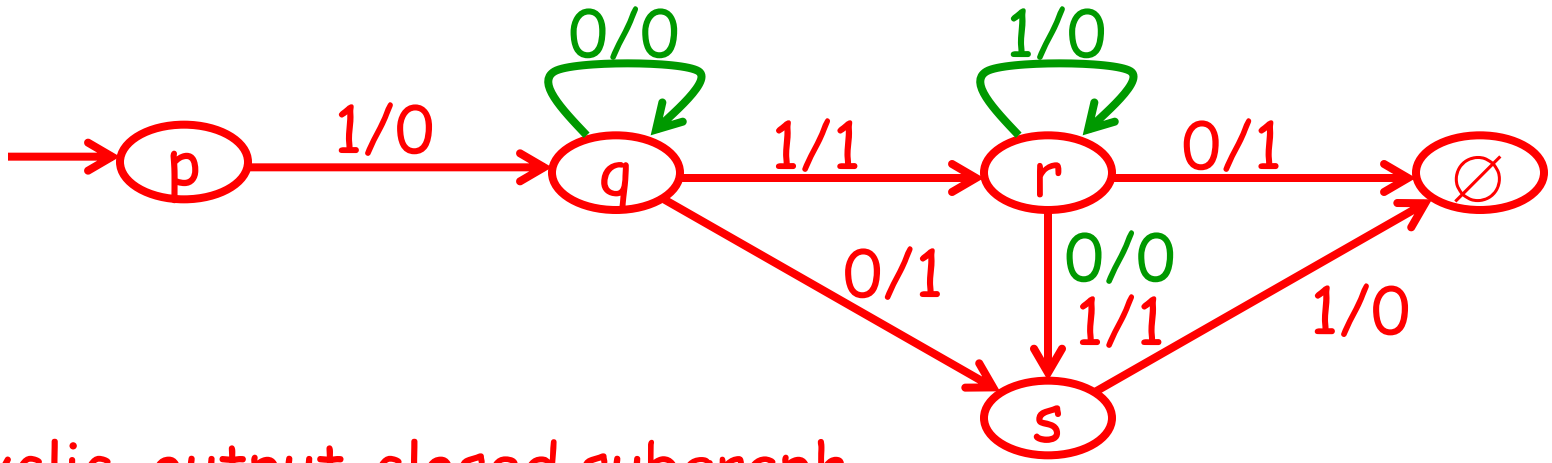
(In other words, the controller always knows the state of the plant.)



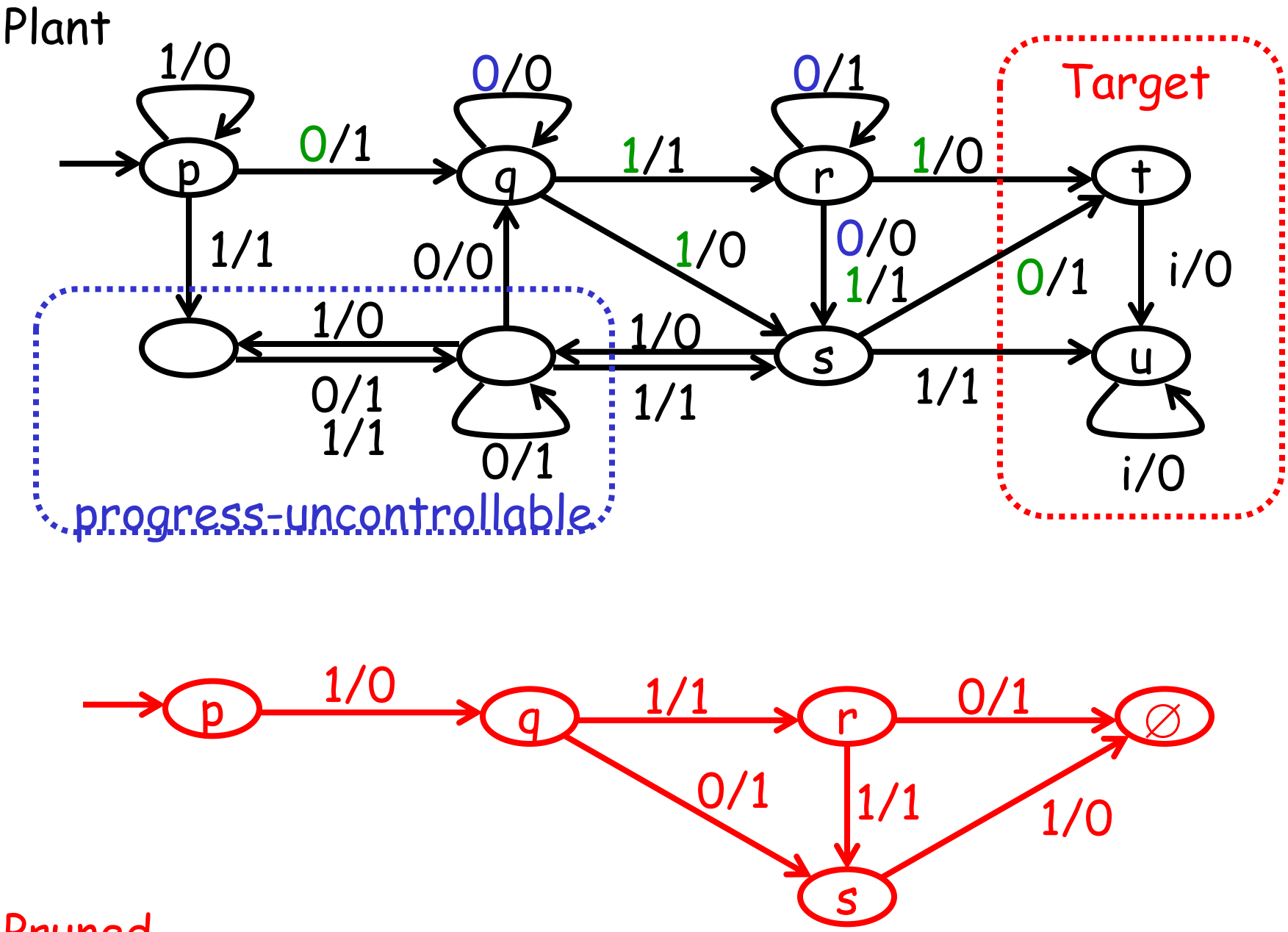
Output-deterministic !

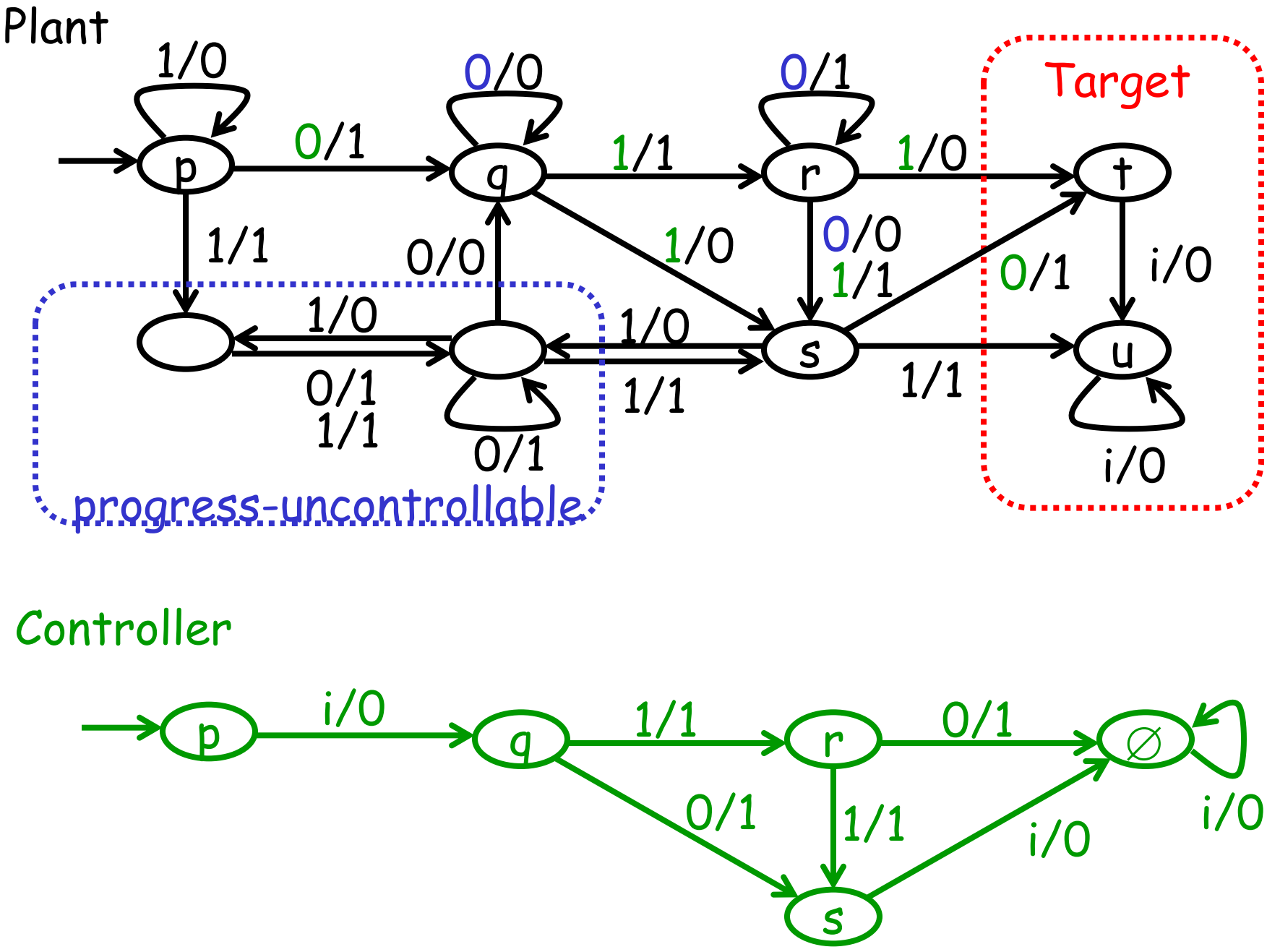

Plant
Target
progress-uncontrollable
p
q
r
s
t
u
1/0
0/1
0/0
1/1
i/0
∅
Safe outputs



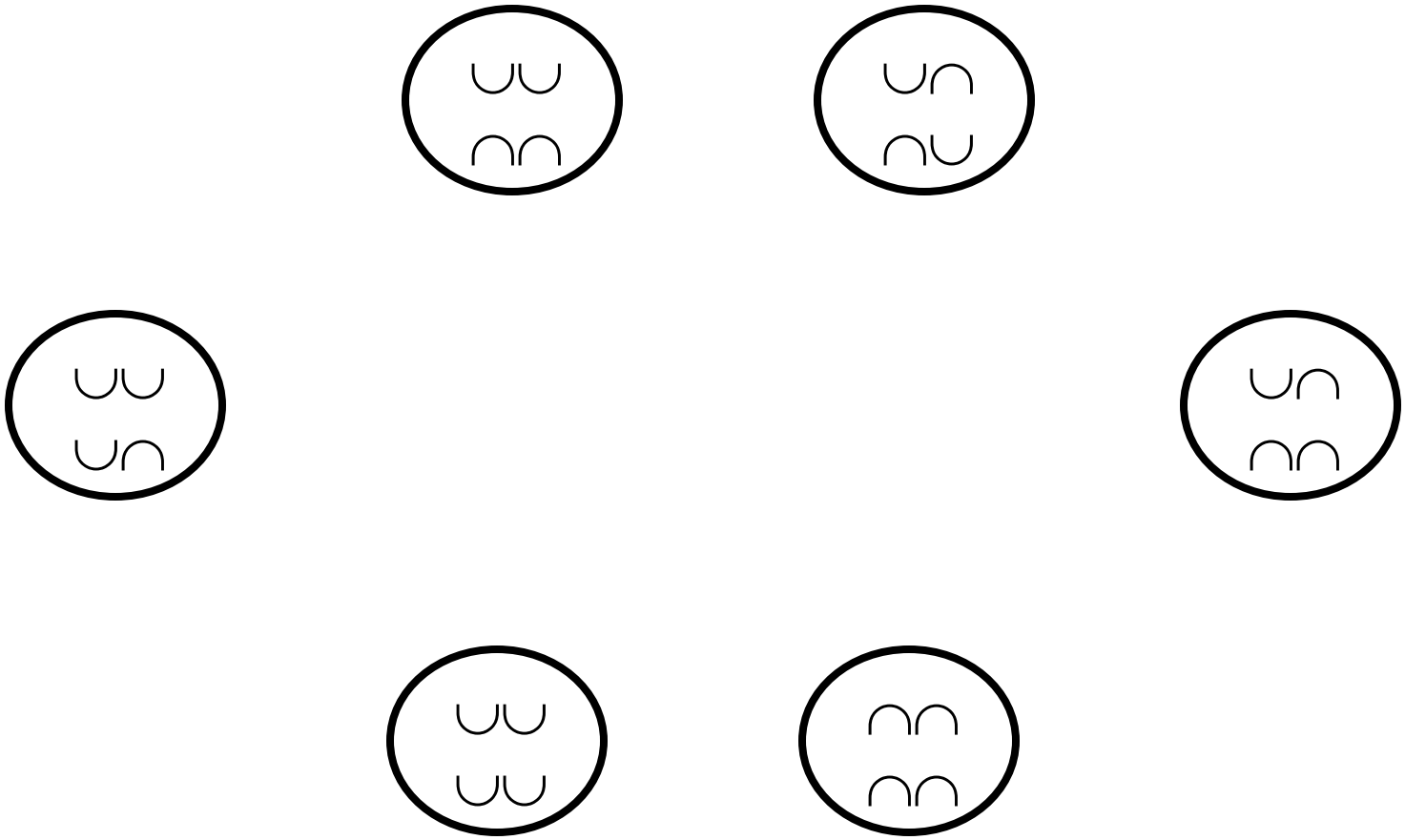


Acyclic, output-closed subgraph

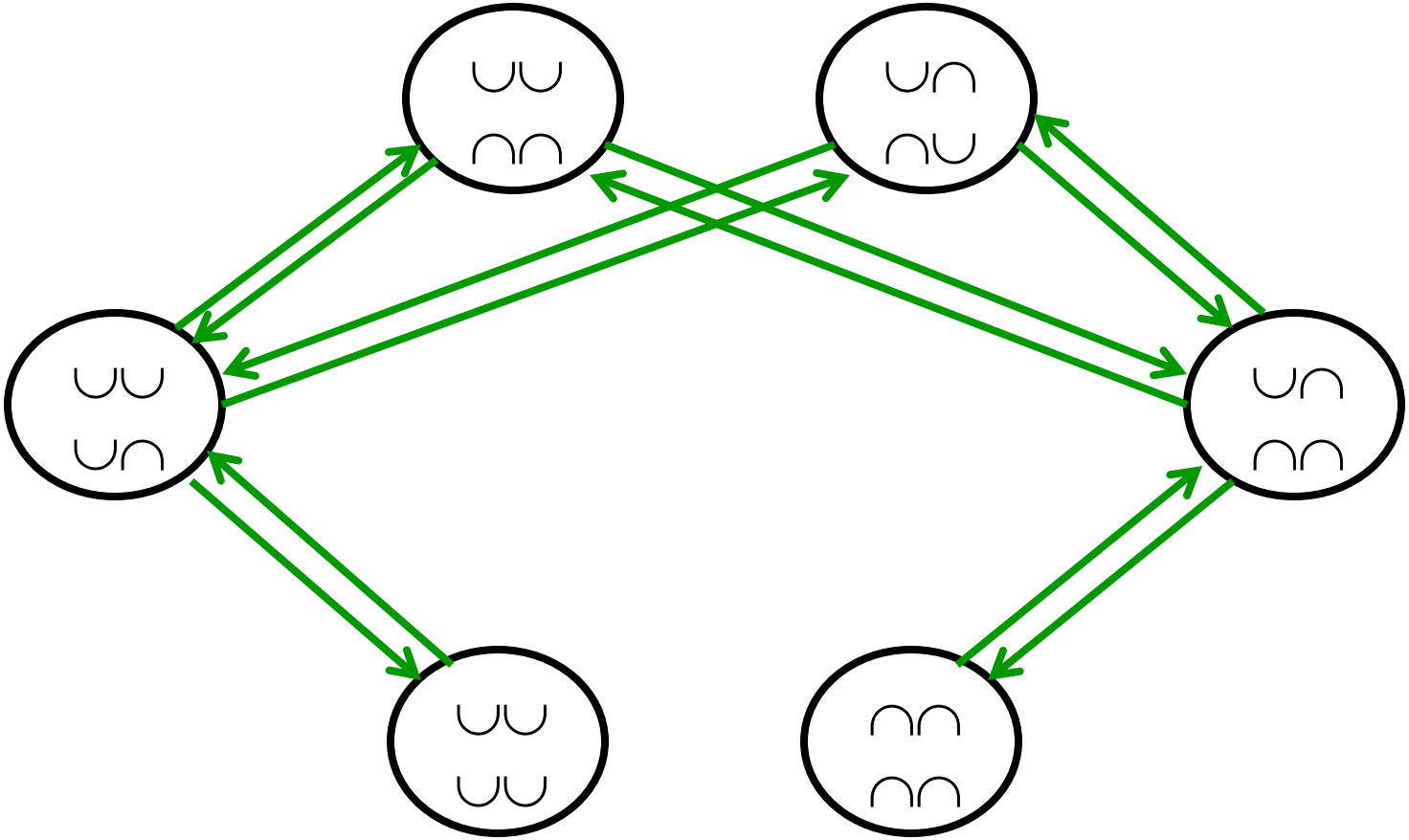

Plant
1/0
0/1
0/0
1/1
0/1
1/0
Target
p
q
r
t
1/1
0/0
1/0
0/0
1/1
0/1
i/0
1/0
1/0
s
u
0/1
1/1
1/1
0/1
1/1
i/0
progress-uncontrollable
p
1/0
q
1/1
r
0/1
∅
0/1
1/1
1/0
s
Pruned


Plant
1/0
0/1
0/0
0/1
1/1
1/0
Target
p
q
r
t
1/1
0/0
1/0
0/0
1/1
0/1
i/0
1/0
1/0
s
u
0/1
1/1
1/1
0/1
1/1
i/0
progress-uncontrollable
Controller
i/0
1/1
0/1
p
q
r
∅
0/1
1/1
i/0
i/0
s

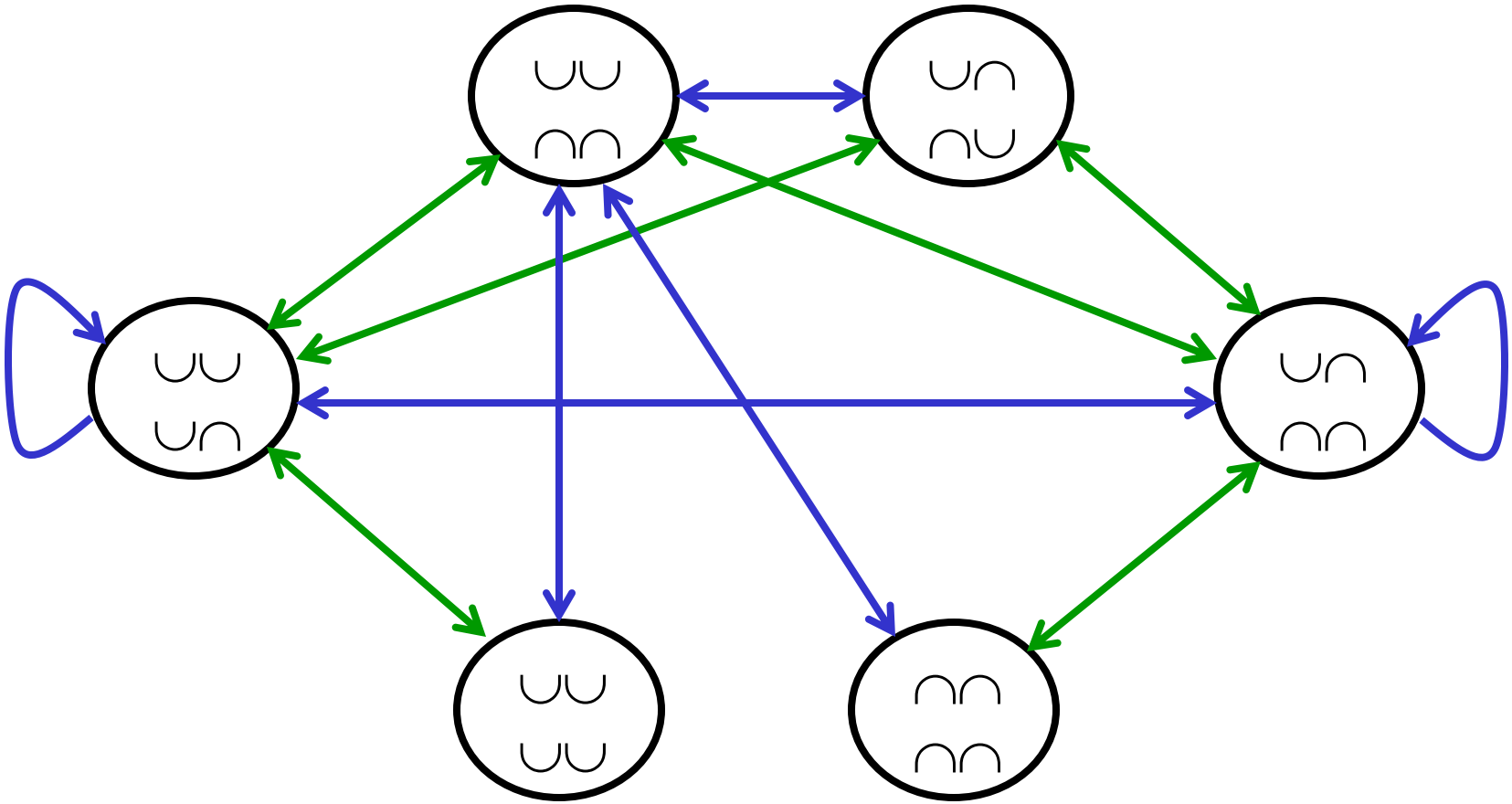
A Game Graph



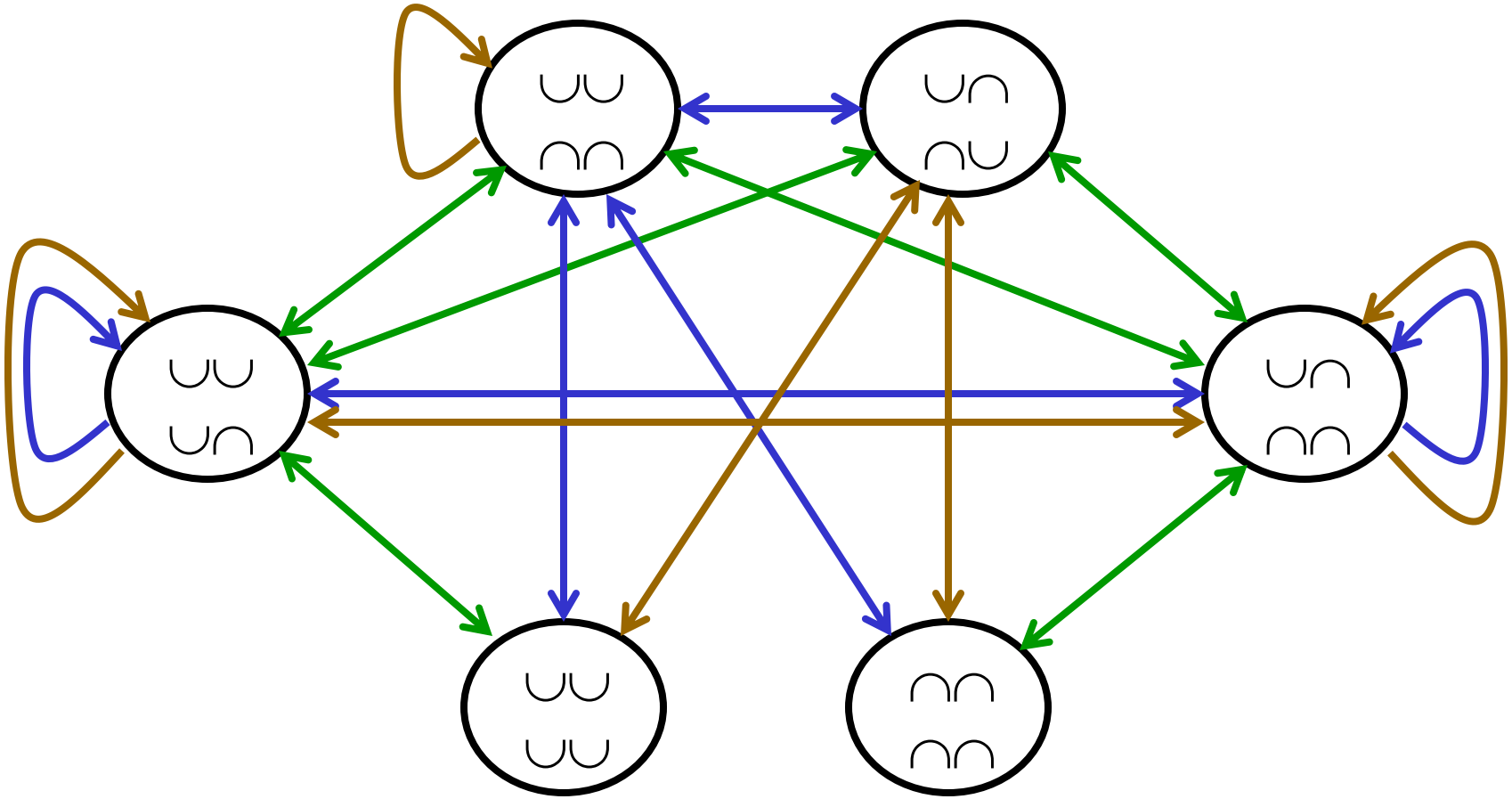
green: turn-1 / •



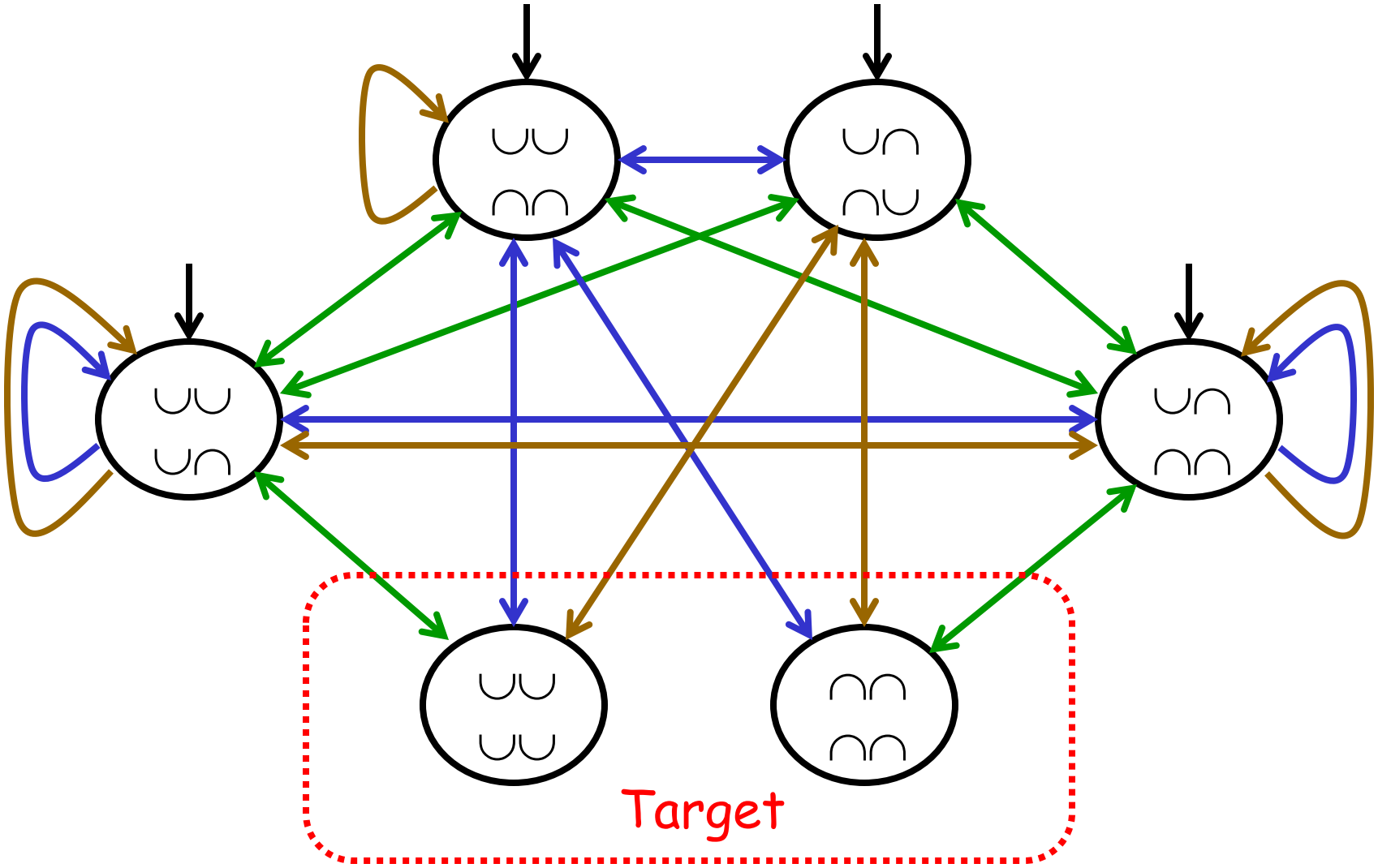
green: turn-1 / •
blue: turn-2-adjacent / •

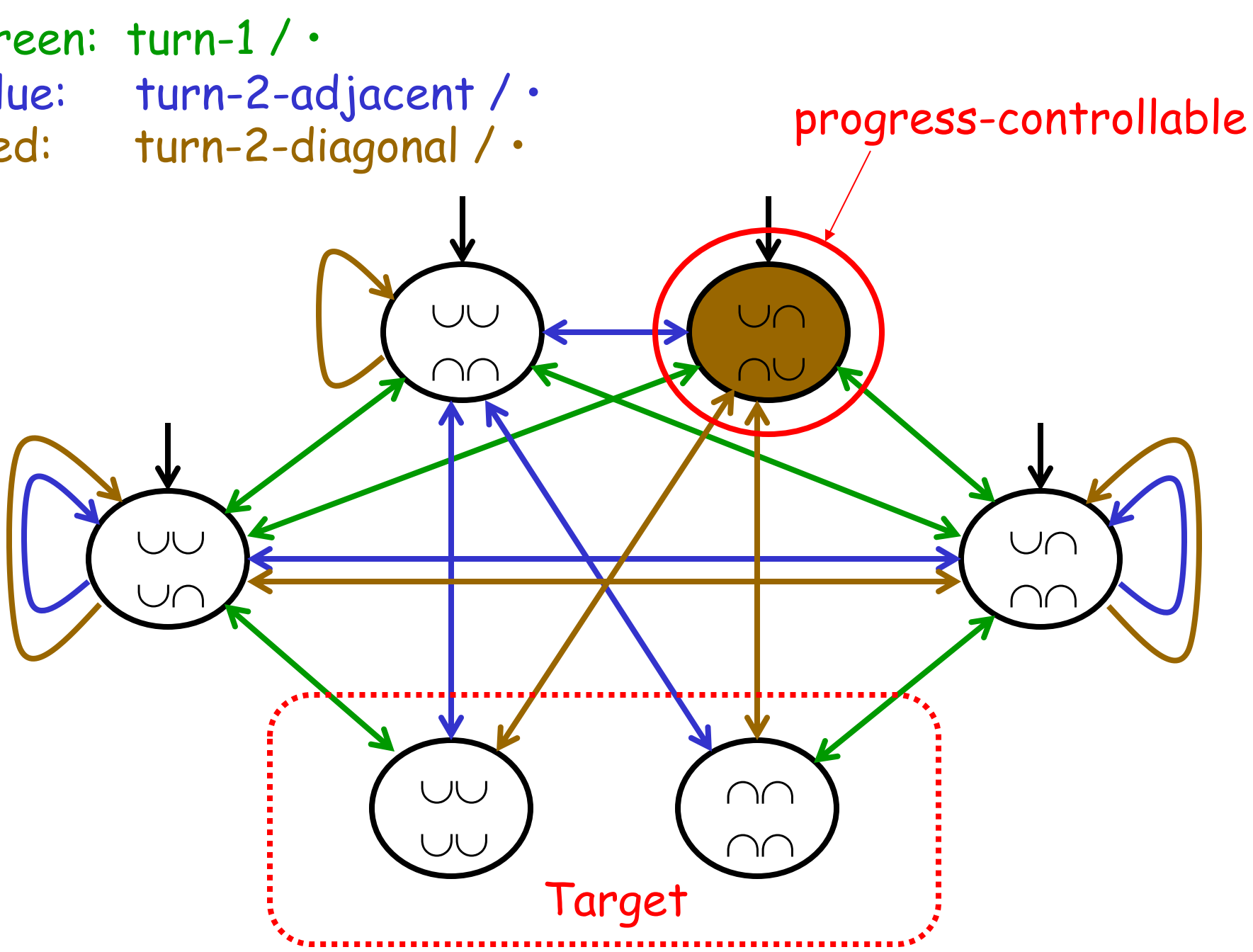


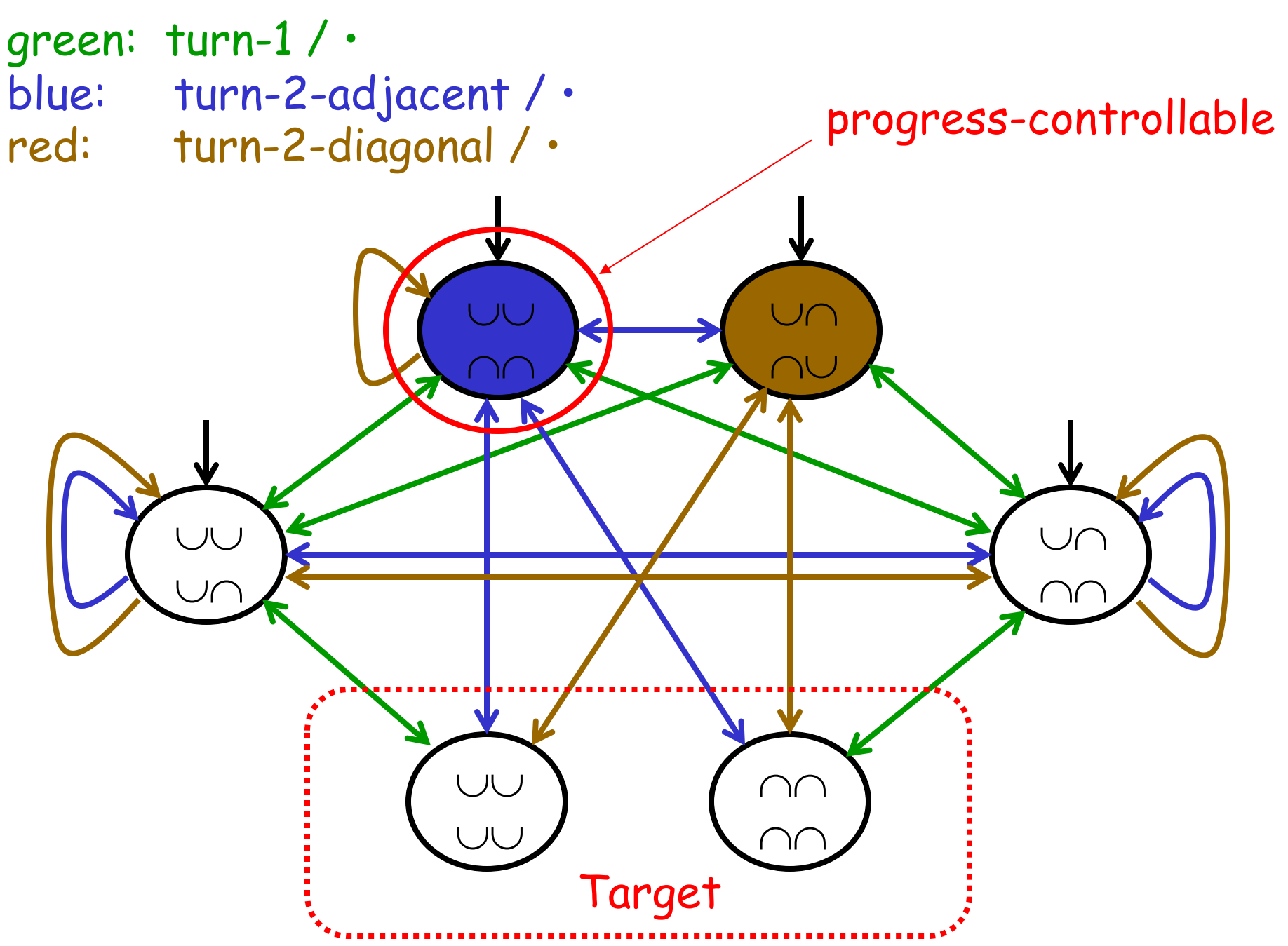
green: turn-1 / •
blue: turn-2-adjacent / •
red: turn-2-diagonal / •

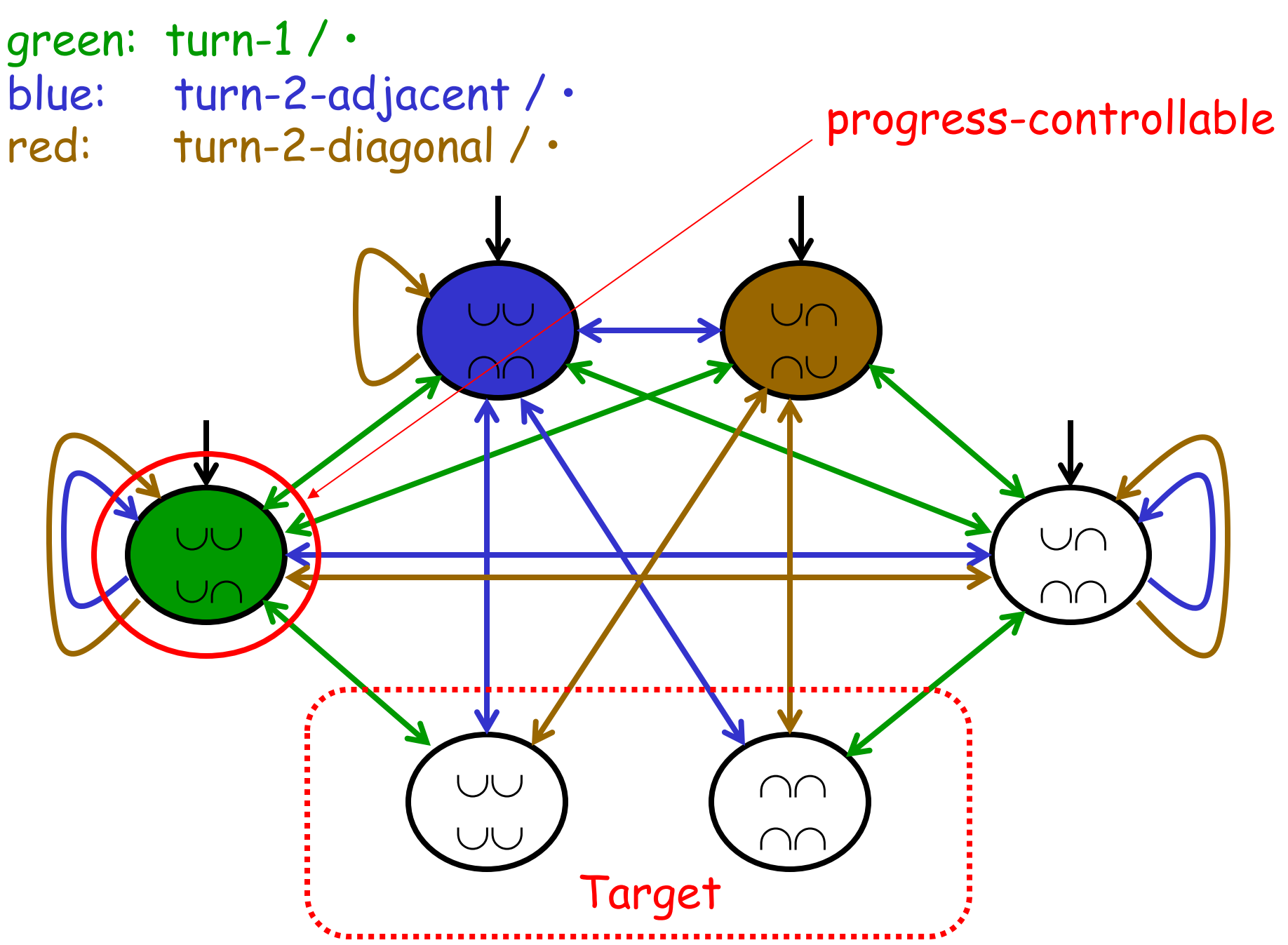


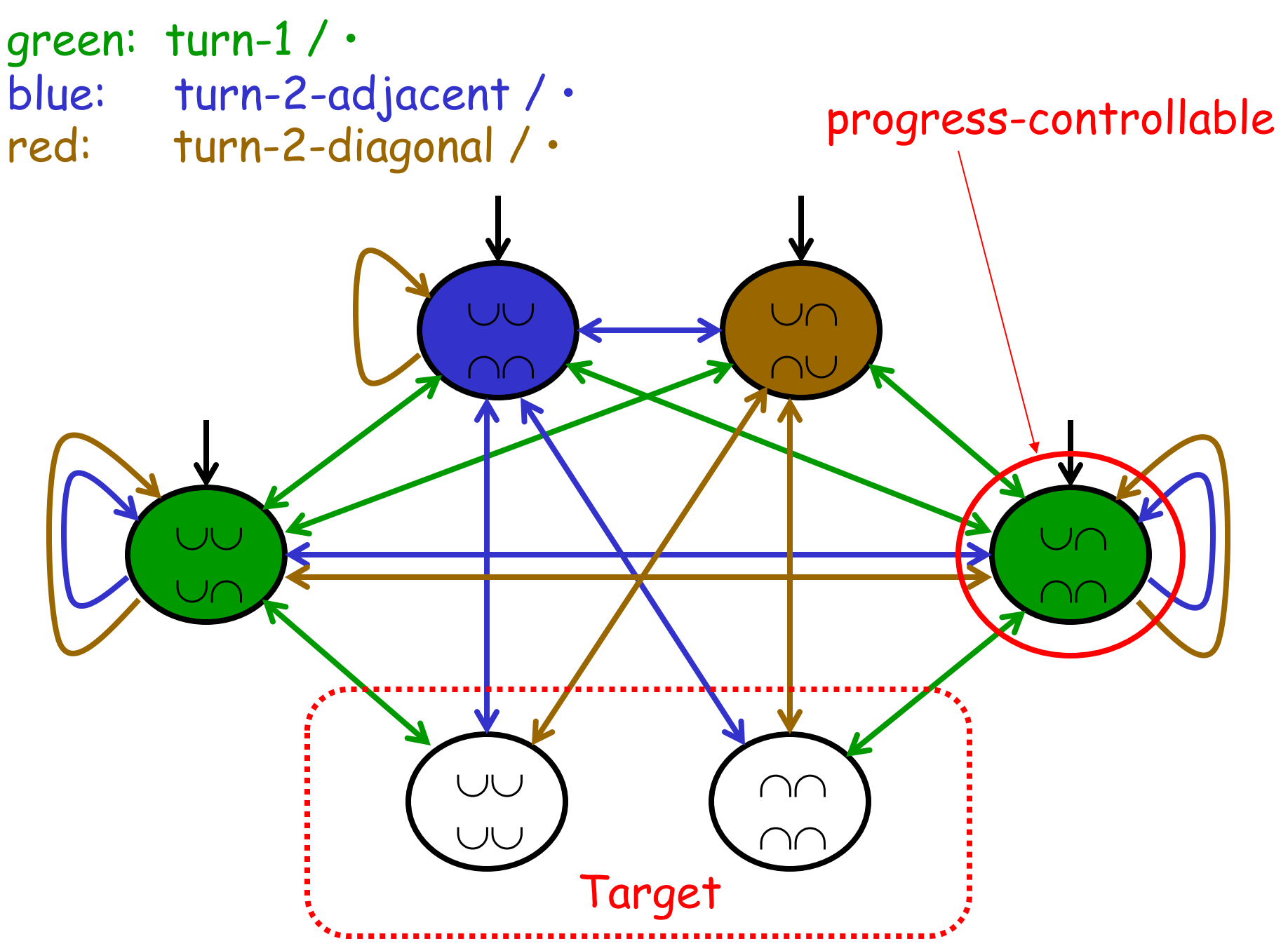
green: turn-1 / •
blue: turn-2-adjacent / •
red: turn-2-diagonal / •




green: turn-1 / •
blue: turn-2-adjacent / •
red: turn-2-diagonal / •
progress-controllable
Target

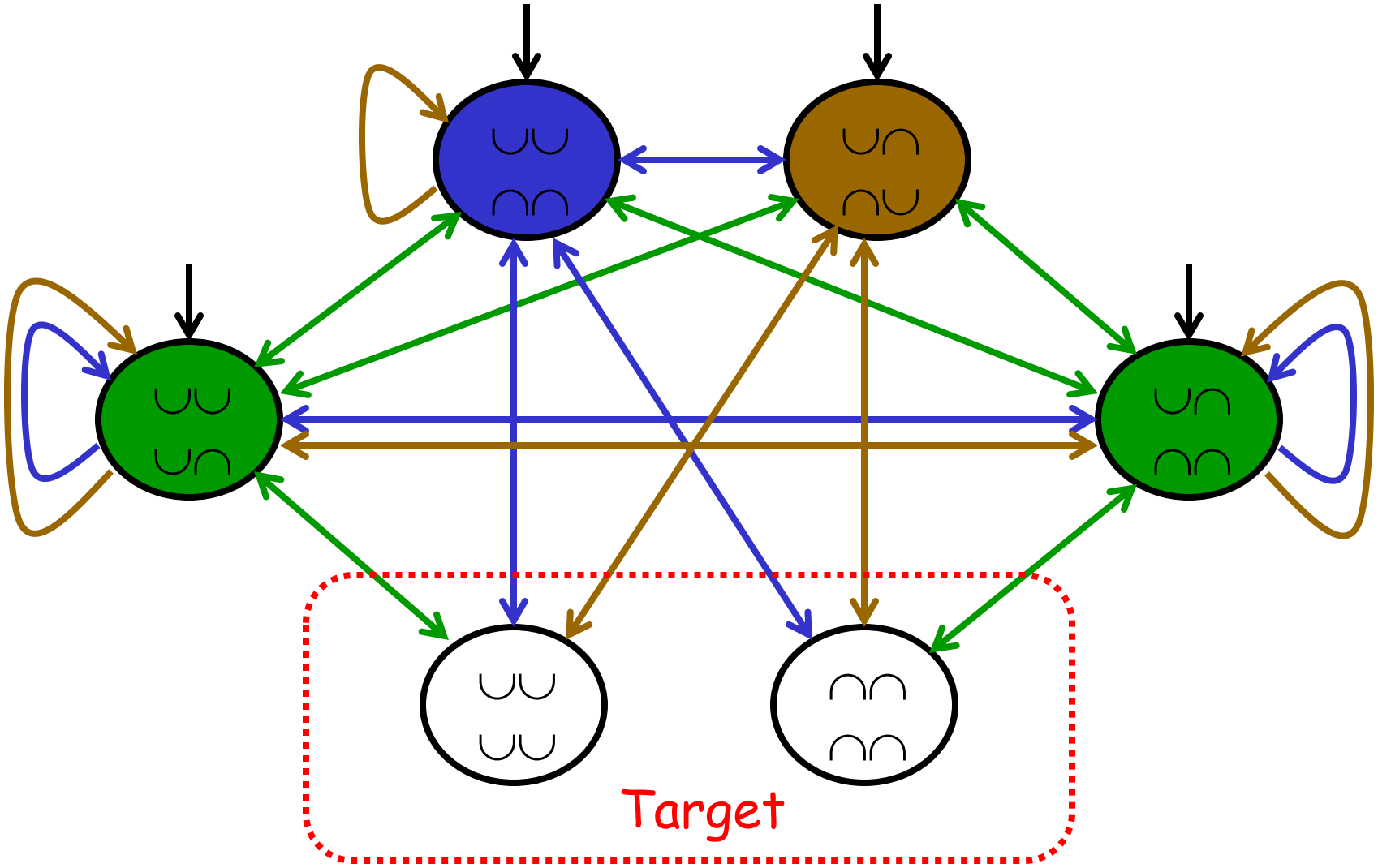

green: turn-1 / •
blue: turn-2-adjacent / •
red: turn-2-diagonal / •
progress-controllable
Target


green: turn-1 / •
blue: turn-2-adjacent / •
red: turn-2-diagonal / •
progress-controllable
Target


green: turn-1 / •
blue: turn-2-adjacent / •
red: turn-2-diagonal / •
progress-controllable
Target

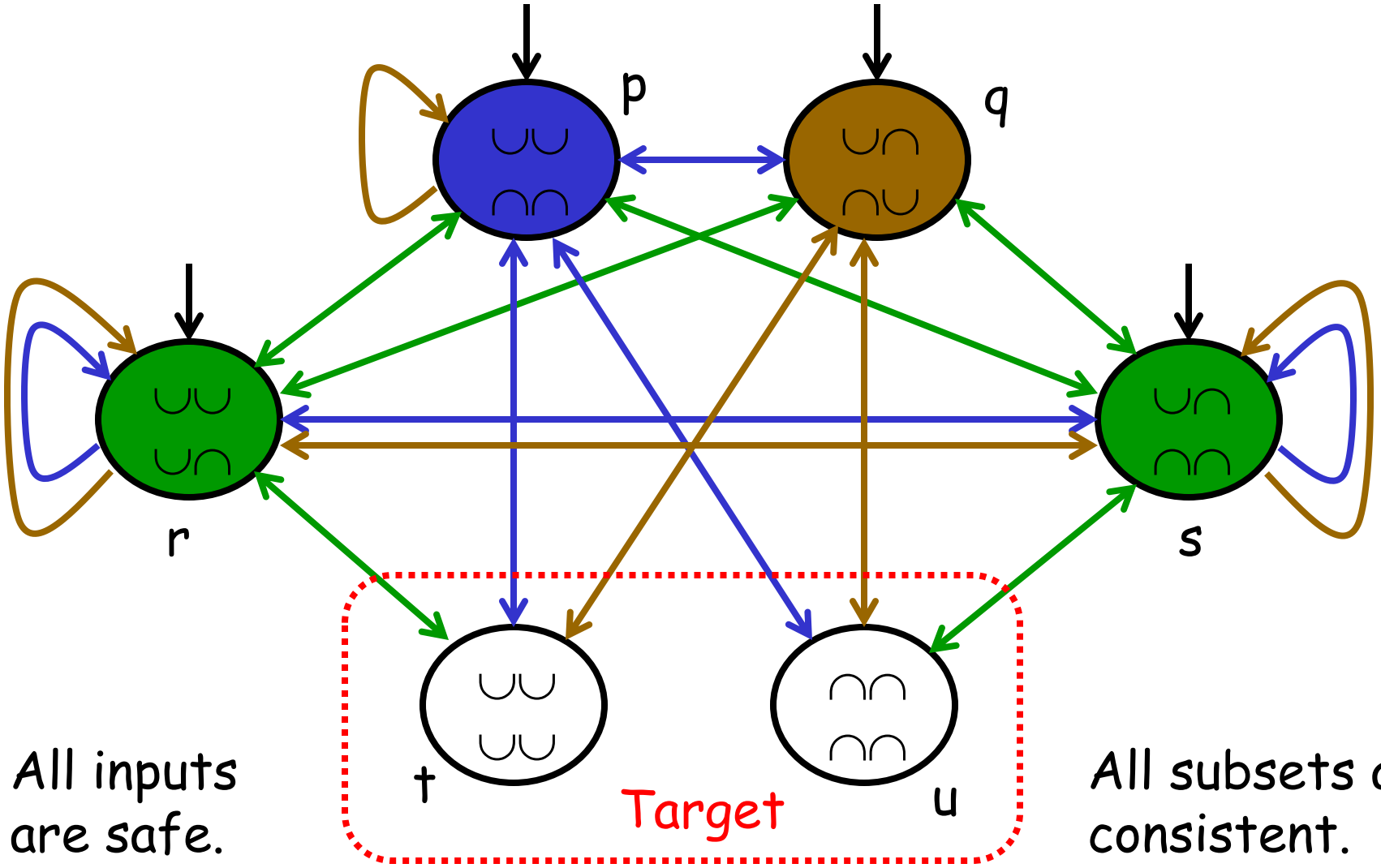
green: turn-1 / •
blue: turn-2-adjacent / •
red: turn-2-diagonal / •

All states are progress-controllable.



green: turn-1 / •
blue: turn-2-adjacent / •
red: turn-2-diagonal / •

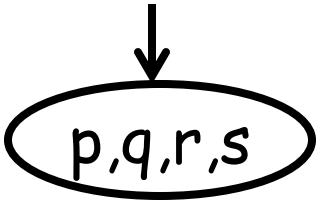
All states are progress-controllable.



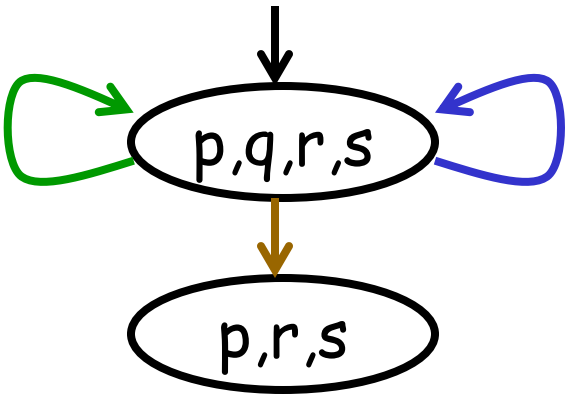
All inputs are safe.

All subsets are consistent.

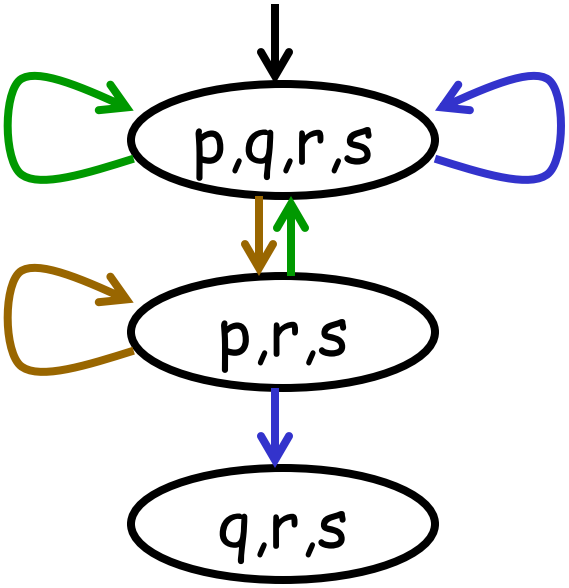
Determinization



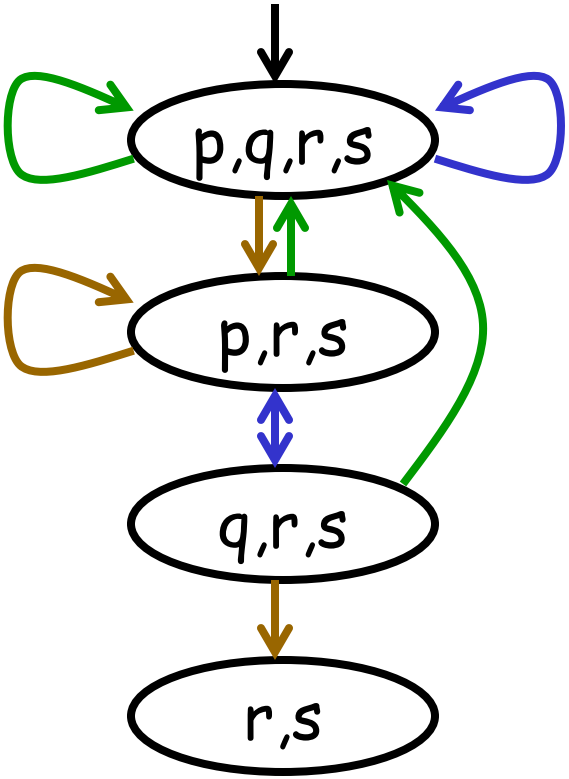
Determinization



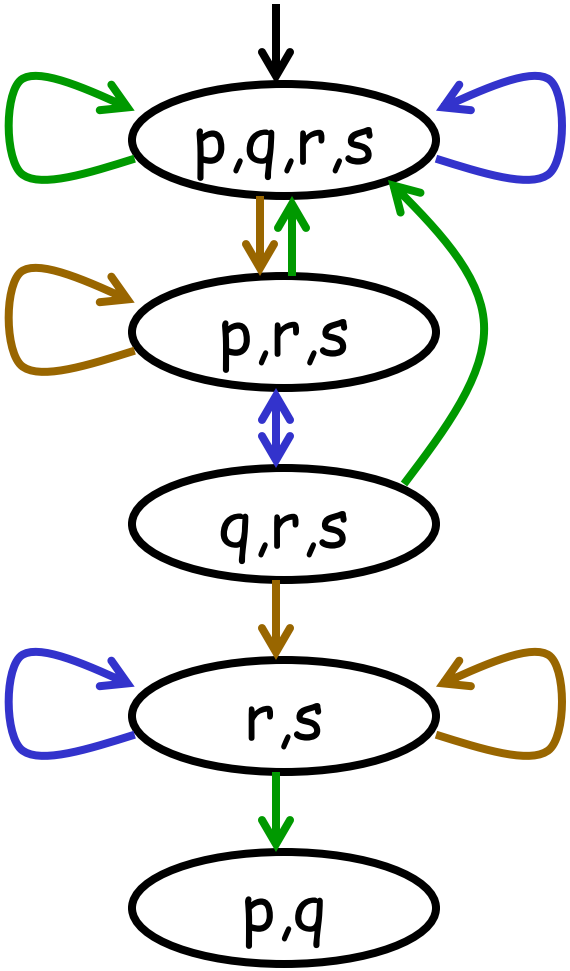
Determinization



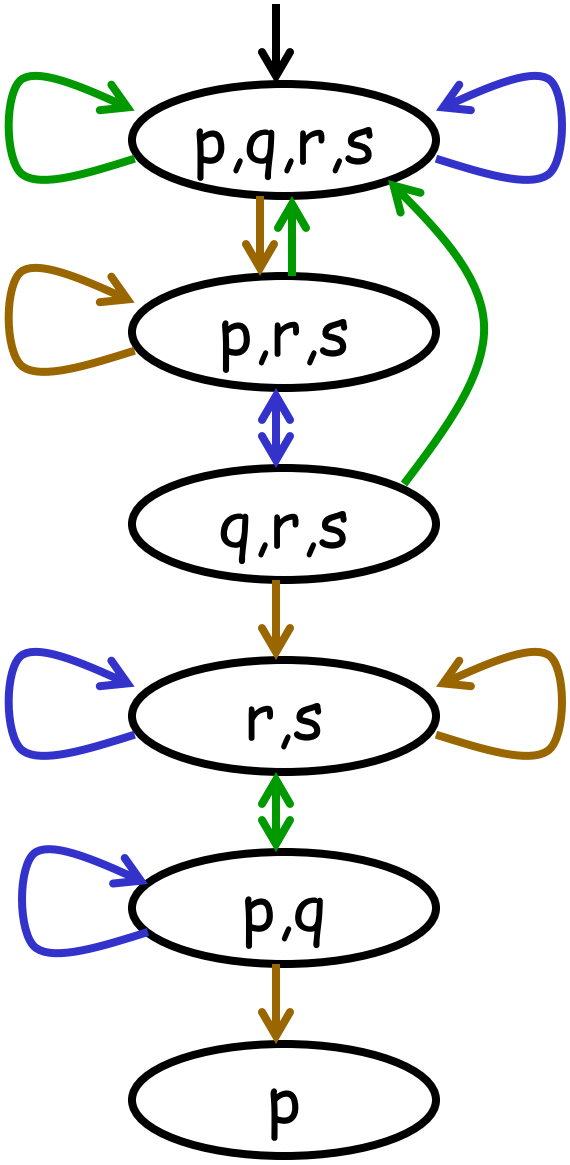
Determinization



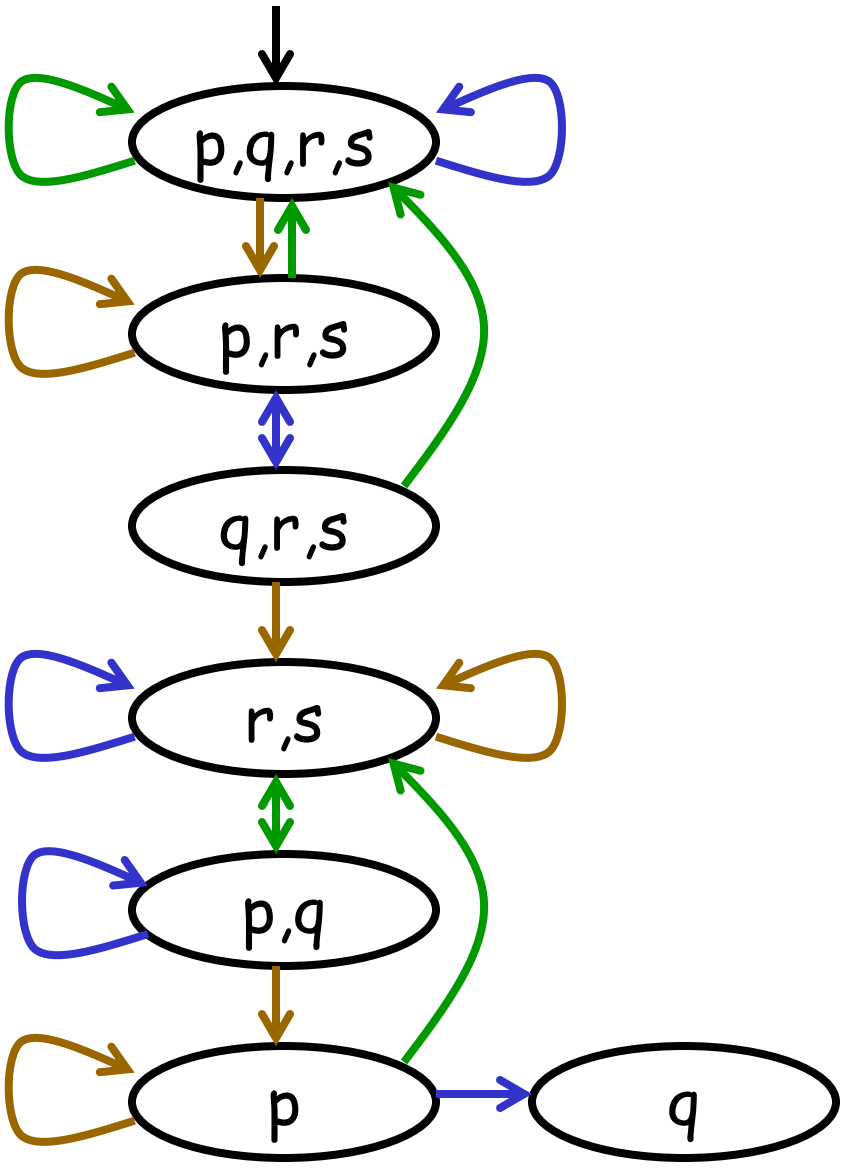
Determinization



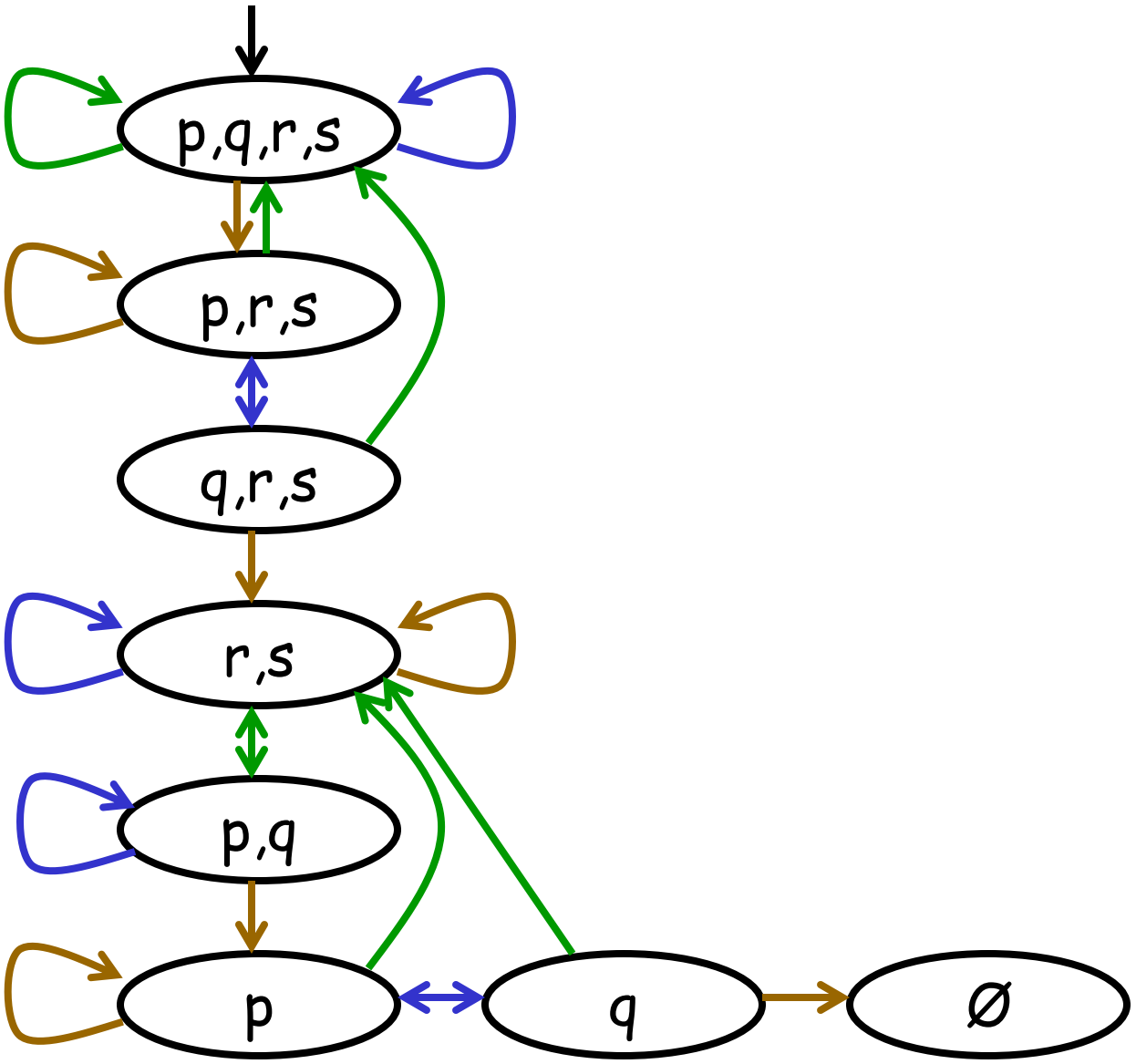
Determinization



Determinization



Determinization



Controller