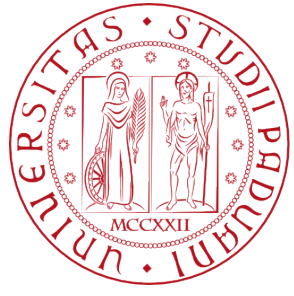
Automi e Linguaggi Formali

a.a. 2017/2018

LT in Informatica
22 Marzo 2018



Università
degli Studi
di Padova

Primo compitino

Lunedì 9 Aprile – ore 13:00

Aule LuM250 e LuF1

- La lista su uniweb è aperta
- Si chiude Sabato 7 aprile
- Domenica 8 verrà pubblicata sul moodle la ripartizione tra le due aule degli iscritti

Esercitazioni in laboratorio

- **26 Aprile** e **3 Maggio**, 12:30-14:30, LabP140
- costruzione di un parser per un linguaggio di programmazione
- e di un traduttore verso il linguaggio C / C++
- usando il generatore automatico di parser ANTLR 4
- ANTLR è scritto in Java, ma genera parser anche in altri linguaggi di programmazione:
  - C#, Python, JavaScript, Go, C++, Swift

Quale linguaggio preferite usare?

Go to:

http://lfb.io/phlof

or

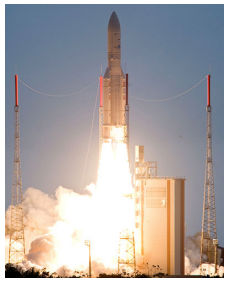
Scan this QR code:

Click to enlarge

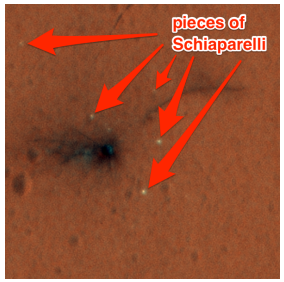
Gli errori di programmazione . . .

. . . possono costare molto cari:

Therac 25
(1985-87, sei incidenti gravi o mortali)



Esplosione dell'Ariane 5
(1996, 500 milioni $)



Schianto del Lander Schiaparelli ? (2016)

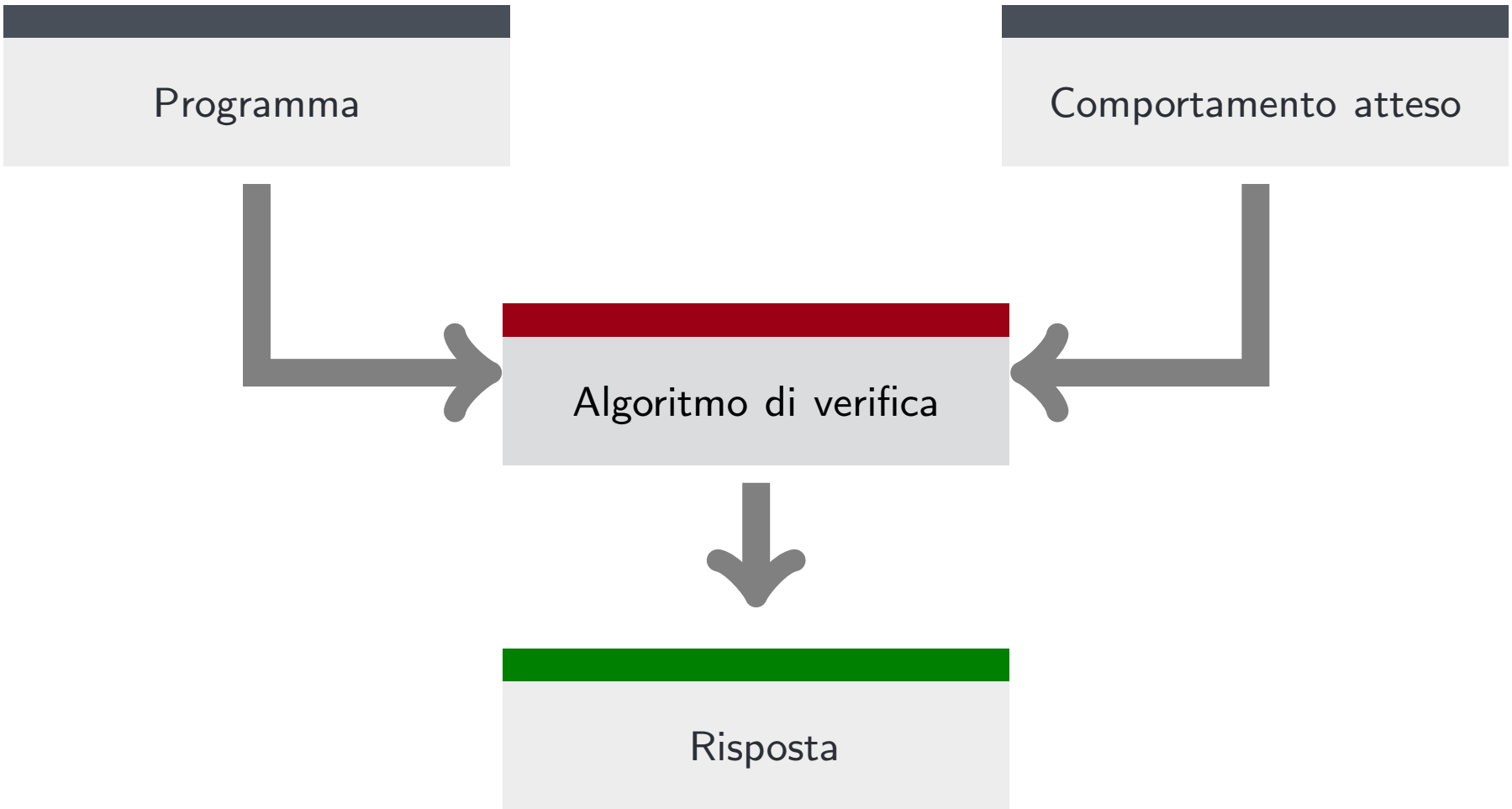
I limiti del testing

- Un modo per trovare errori nei programmi è fare dei test
- I test coprono solo un sottoinsieme dei possibili input
- Riescono a dimostrare l'esistenza di errori, ma non la loro assenza
- Un programma non opera in isolamento: il suo comportamento dipende da quello che succede nell'ambiente esterno
  - gli errori causati dall'interazione con l'ambiente o gli utenti sono molto difficili da individuare!

Un modo diverso per trovare gli errori

- Metodi Formali:
  - usare un linguaggio matematico ...
  - ... per descrivere il comportamento corretto ...
  - ... e aiutare il programmatore a scrivere programmi
- Permettono di stabilire la correttezza del programma per tutti i possibili input
- Sono pratica comune nello sviluppo di HW e SW:
  - progettazione di microprocessori (Intel), software critico (NASA), sistemi operativi (Microsoft), ...

Schema di un metodo formale

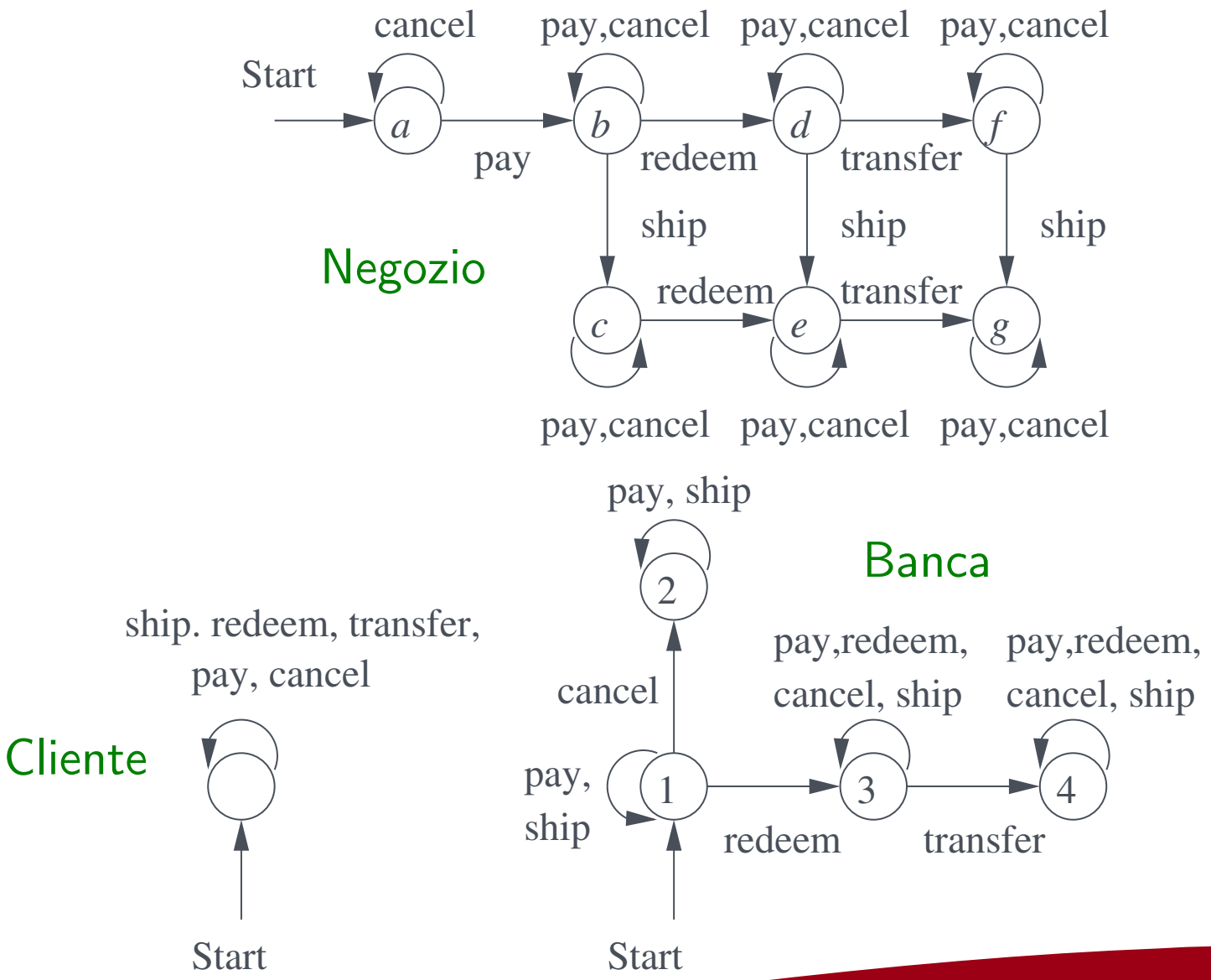


Un sistema di commercio elettronico

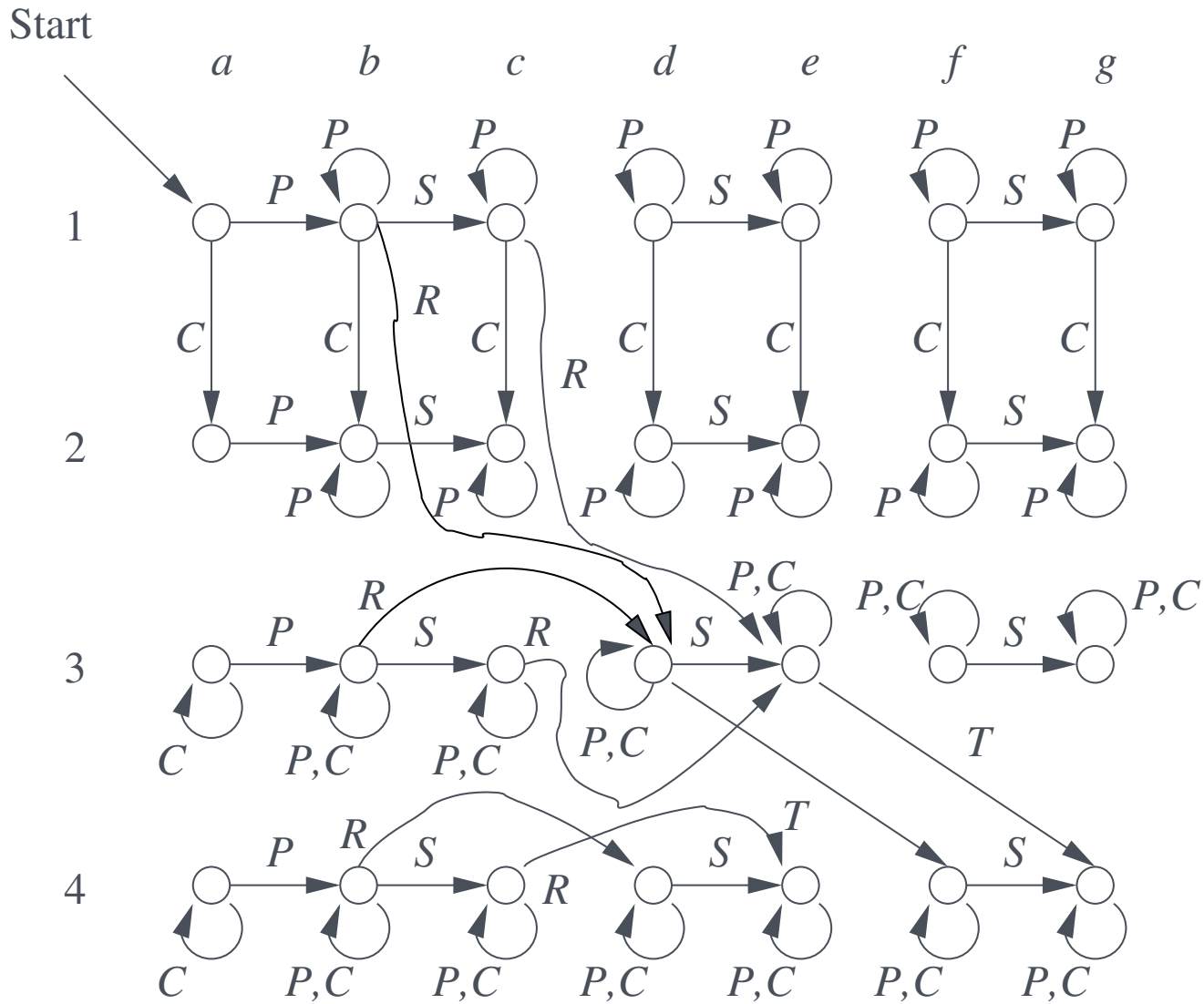
Costruiamo un esempio di commercio elettronico:

- Il cliente paga il negozio con moneta elettronica
- Il cliente può cancellare la moneta elettronica
- Il negozio riceve il pagamento e spedisce il prodotto al cliente
- Per completare il pagamento, il negozio riscatta la moneta elettronica
- La banca controlla la validità della moneta e trasferisce la somma al negozio

Modelliamo l'esempio



Il sistema completo



Comportamento atteso e algoritmo di verifica

- "Il negozio spedisce la merce solo dopo che ha incassato il pagamento"

Comportamento atteso e algoritmo di verifica

- "Il negozio spedisce la merce solo dopo che ha incassato il pagamento"

$$\mathsf{S} = \mathbf{pay} + \mathbf{cancel} + \mathbf{redeem} + \mathbf{transfer} + \mathbf{ship}$$
$$\mathsf{T} = \mathbf{pay} + \mathbf{cancel} + \mathbf{redeem} + \mathbf{transfer}$$
$$\mathsf{R} = \mathsf{T}^* \, \mathbf{transfer} \, \mathsf{T}^* \, \mathbf{ship} \, \mathsf{S}^* + \mathsf{T}^*$$

Comportamento atteso e algoritmo di verifica

- "Il negozio spedisce la merce solo dopo che ha incassato il pagamento"

$$\mathbf{S} = \mathbf{pay} + \mathbf{cancel} + \mathbf{redeem} + \mathbf{transfer} + \mathbf{ship}$$
$$\mathbf{T} = \mathbf{pay} + \mathbf{cancel} + \mathbf{redeem} + \mathbf{transfer}$$
$$\mathbf{R} = \mathbf{T}^* \, \mathbf{transfer} \, \mathbf{T}^* \, \mathbf{ship} \, \mathbf{S}^* + \mathbf{T}^*$$

Per verificare se il sistema S rispetta i requisiti dobbiamo controllare che $L(S) \subseteq L(\mathbf{R})$

Safety Control

EECS 20

Lecture 36 (April 23, 2001)

Tom Henzinger

The Control Problem

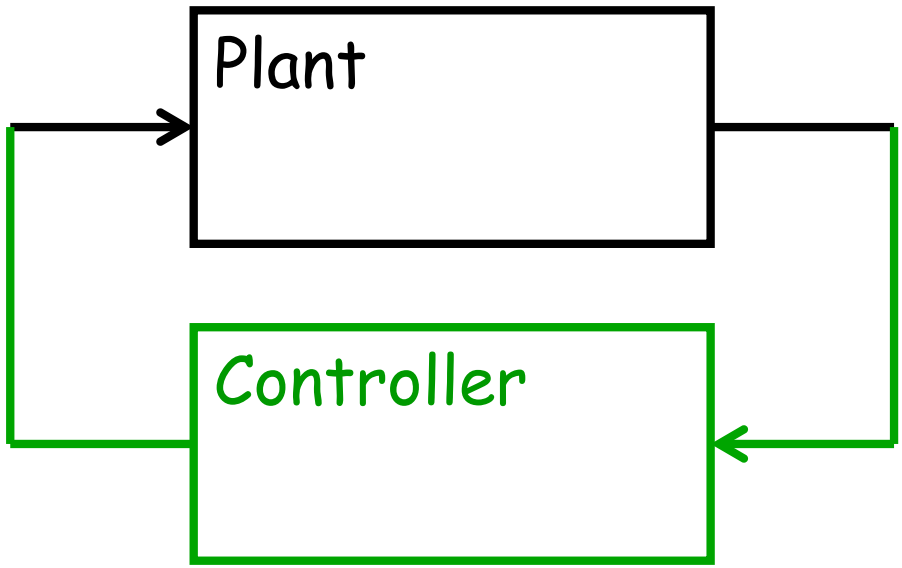
Given

1. 

2. Objective

The Control Problem

Find



such that the composite ("closed-loop") system satisfies the Objective

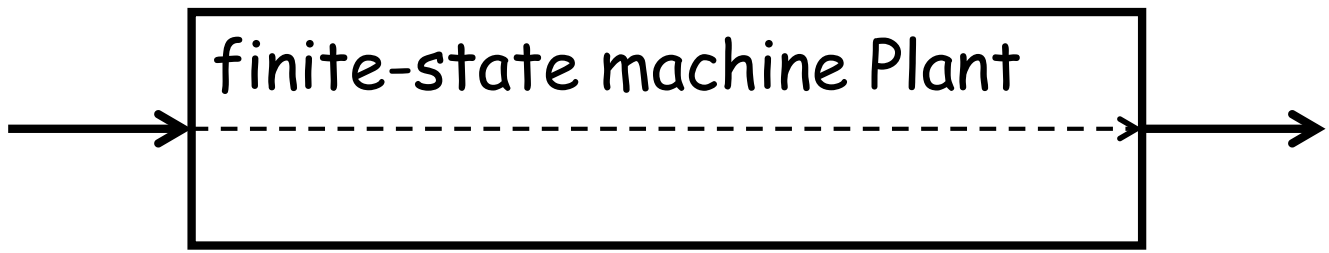
Simplest Finite-State Control Objective:

SAFETY

stay out of a set of undesirable plant states
(the "error" states)

The Finite-State Safety Control Problem

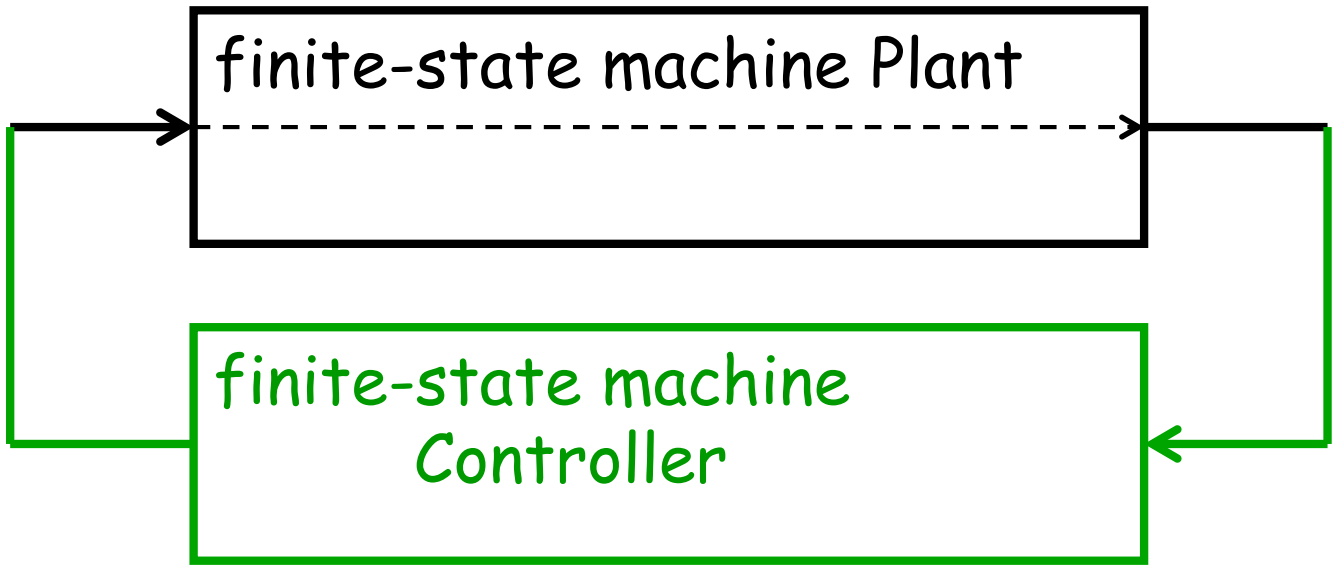
Given

1. 

2. set Error of states of Plant

The Finite-State Safety Control Problem

Find



such that the composite system never enters a state in Error

Step 1:

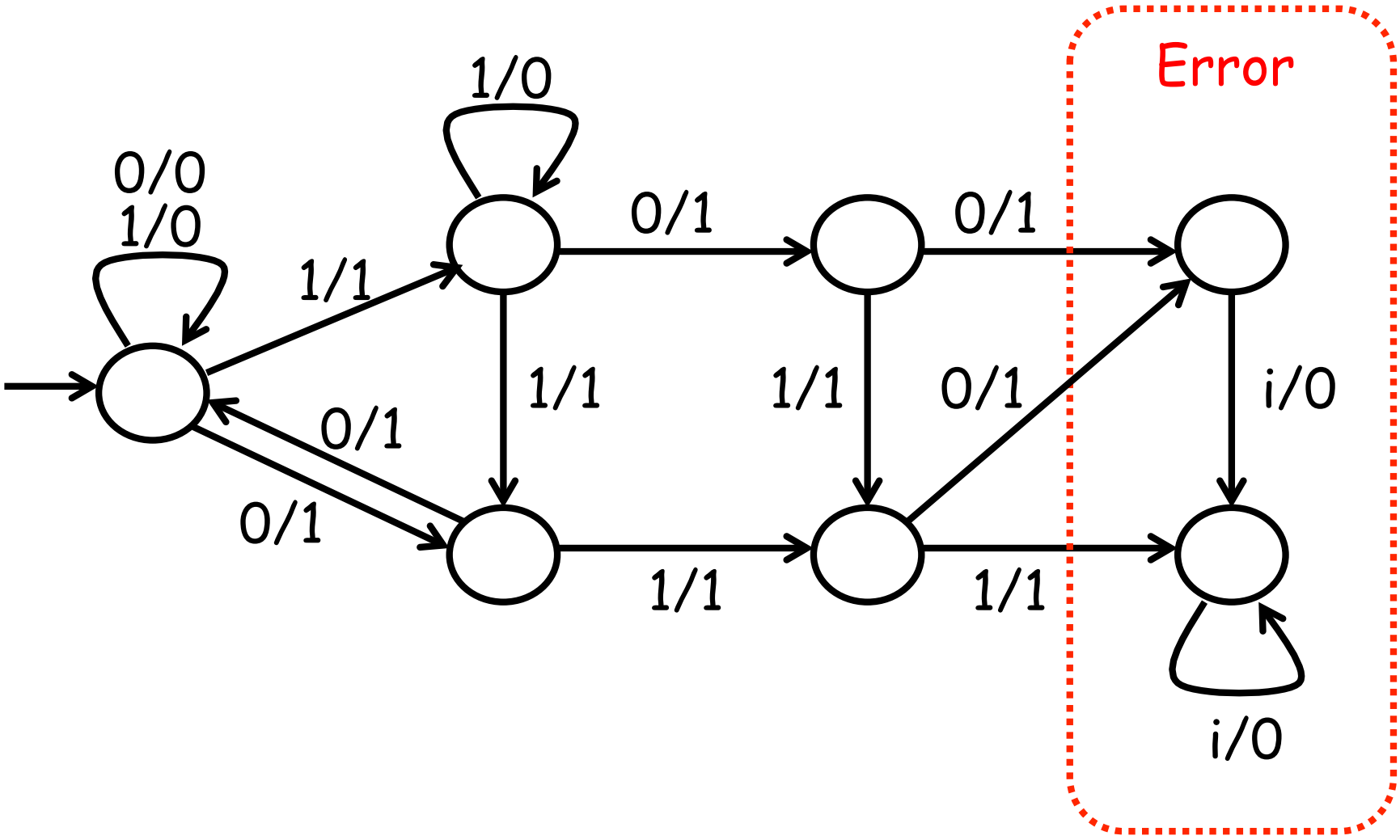
Compute the "uncontrollable" states of Plant

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

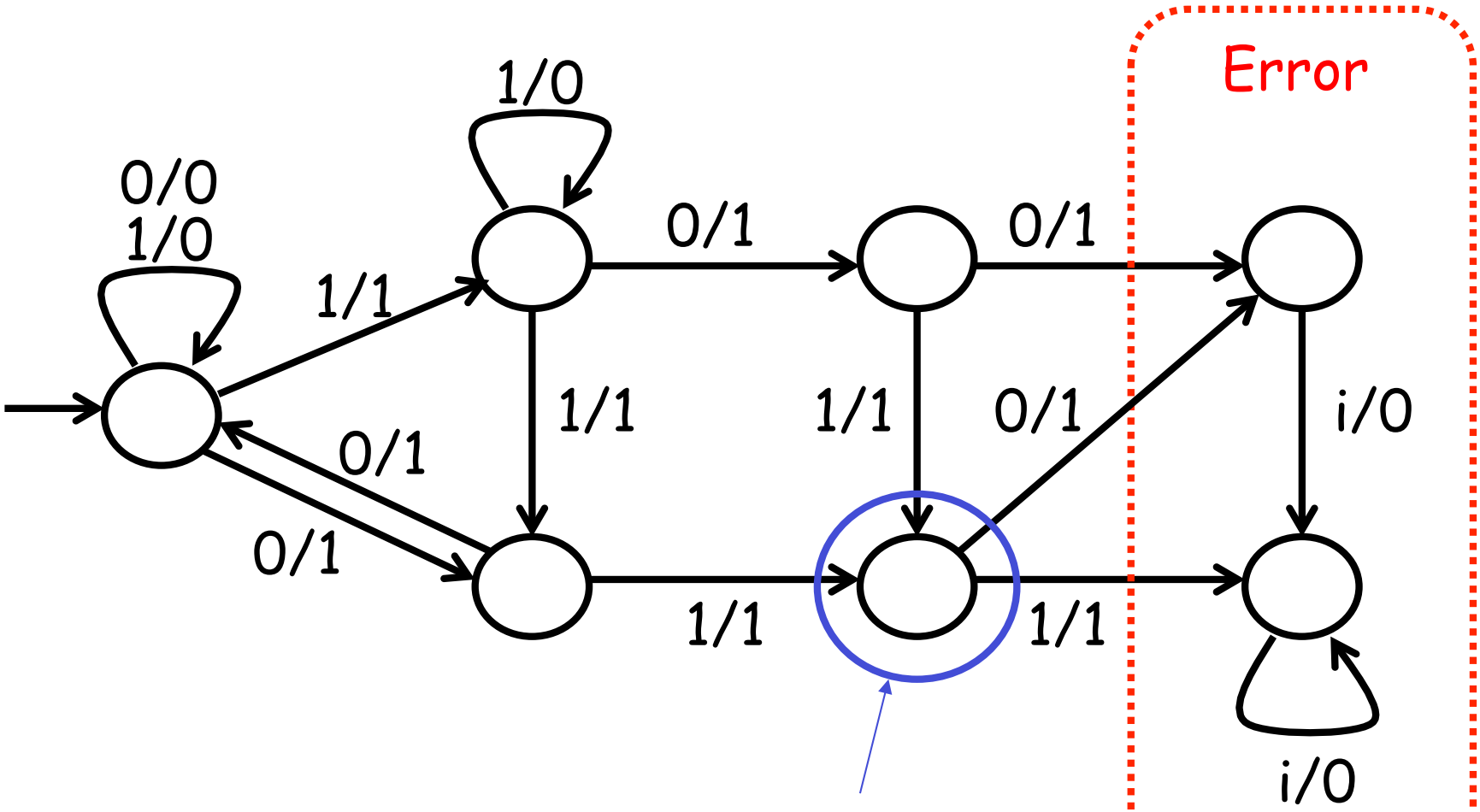
   if for all inputs i
   there exist an uncontrollable state s'
   and an output o
   such that $(s', o) \in$ possibleUpdates (s,i)

   then s is uncontrollable.

Plant



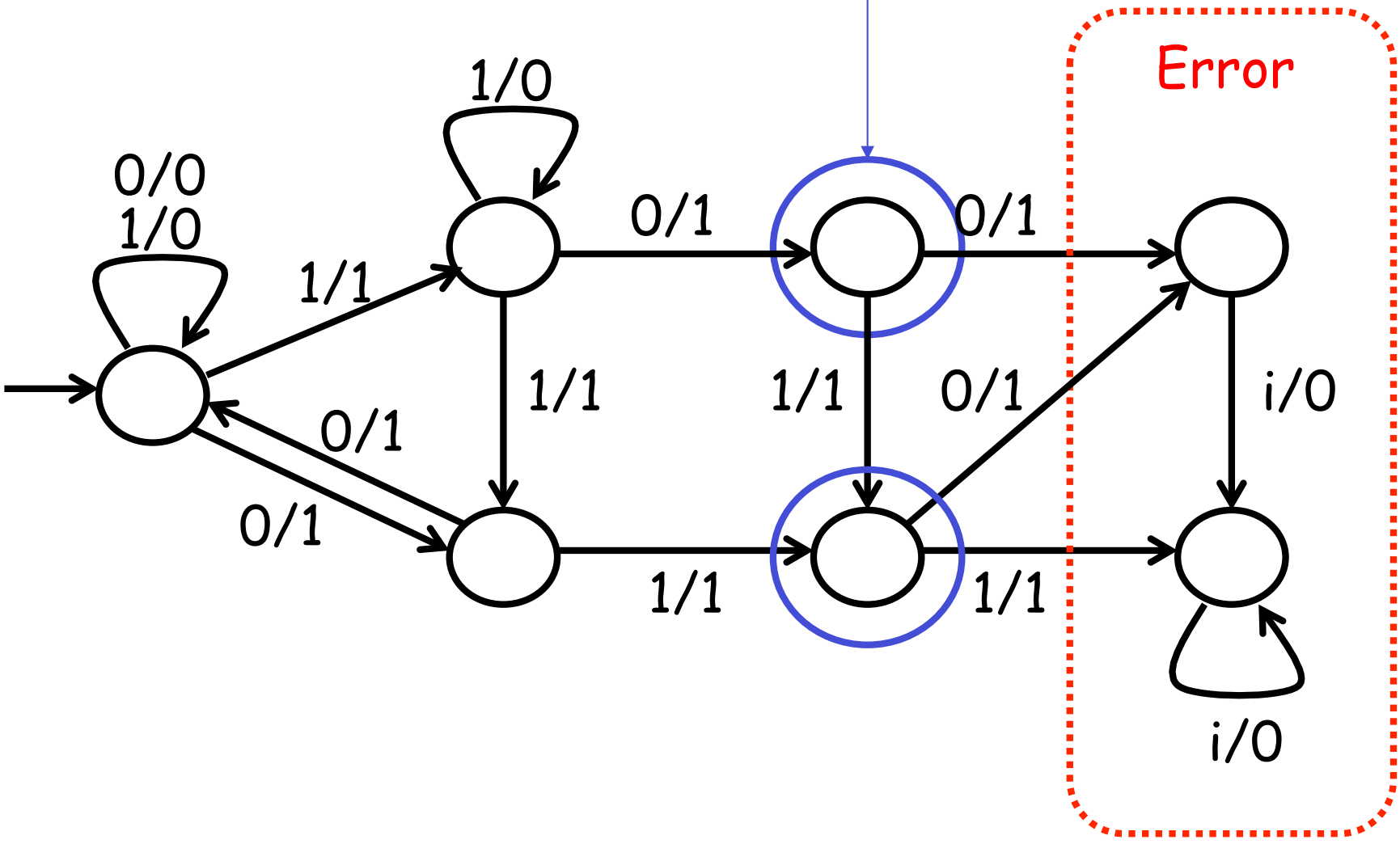
Plant

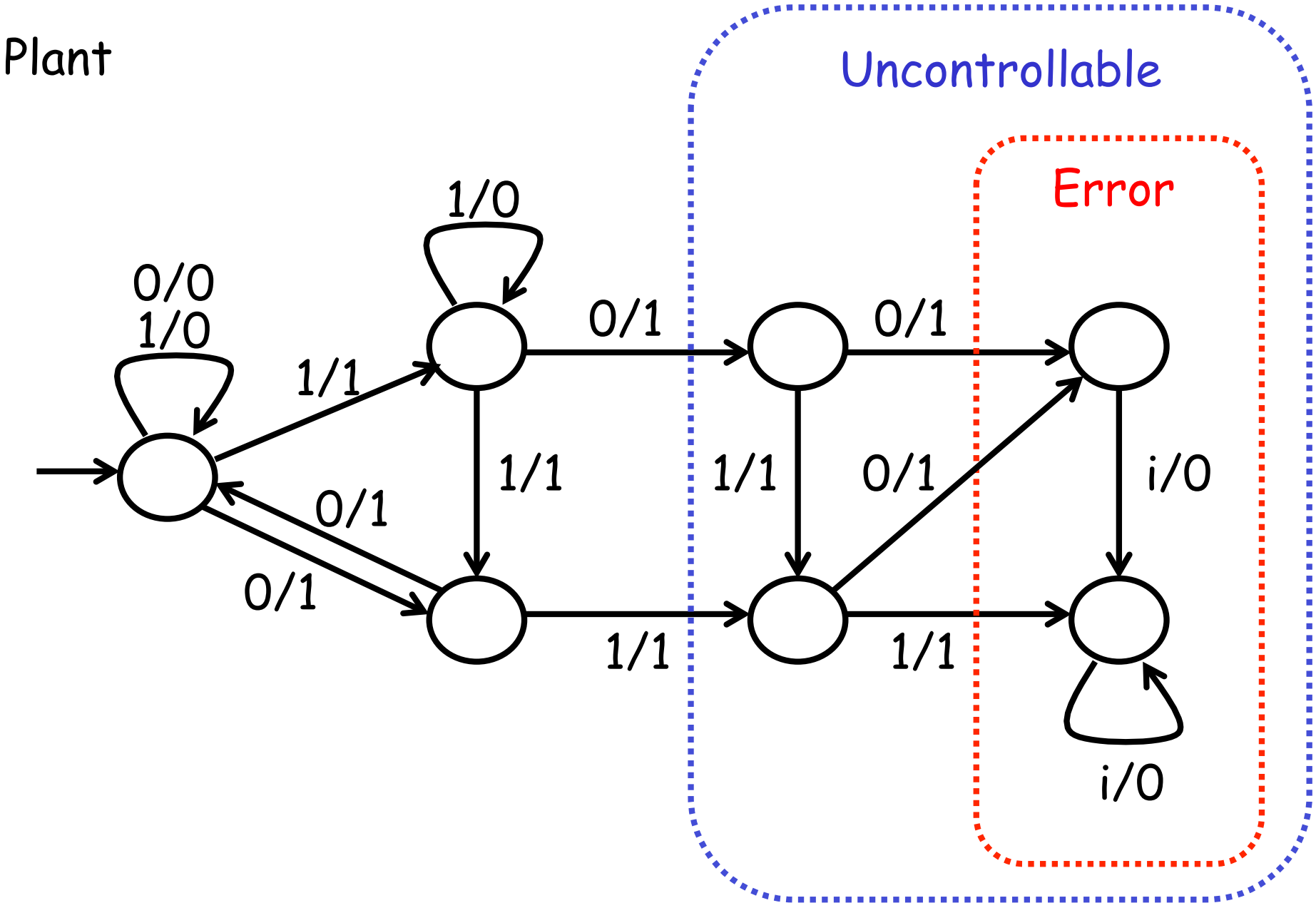


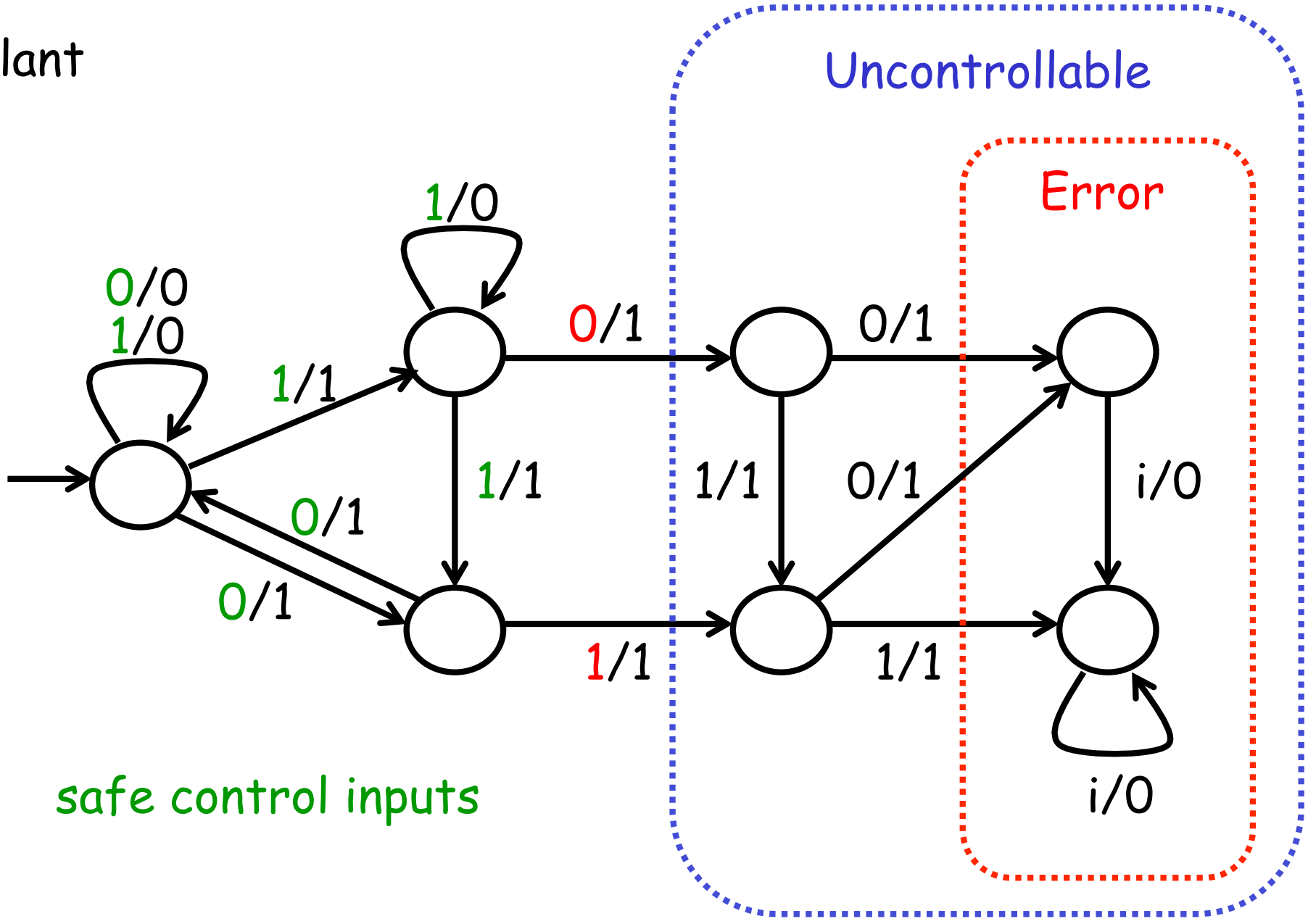
uncontrollable (cannot prevent error state from being entered in 1 transition)

Plant

uncontrollable (cannot prevent error state from being entered in 2 transitions)




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

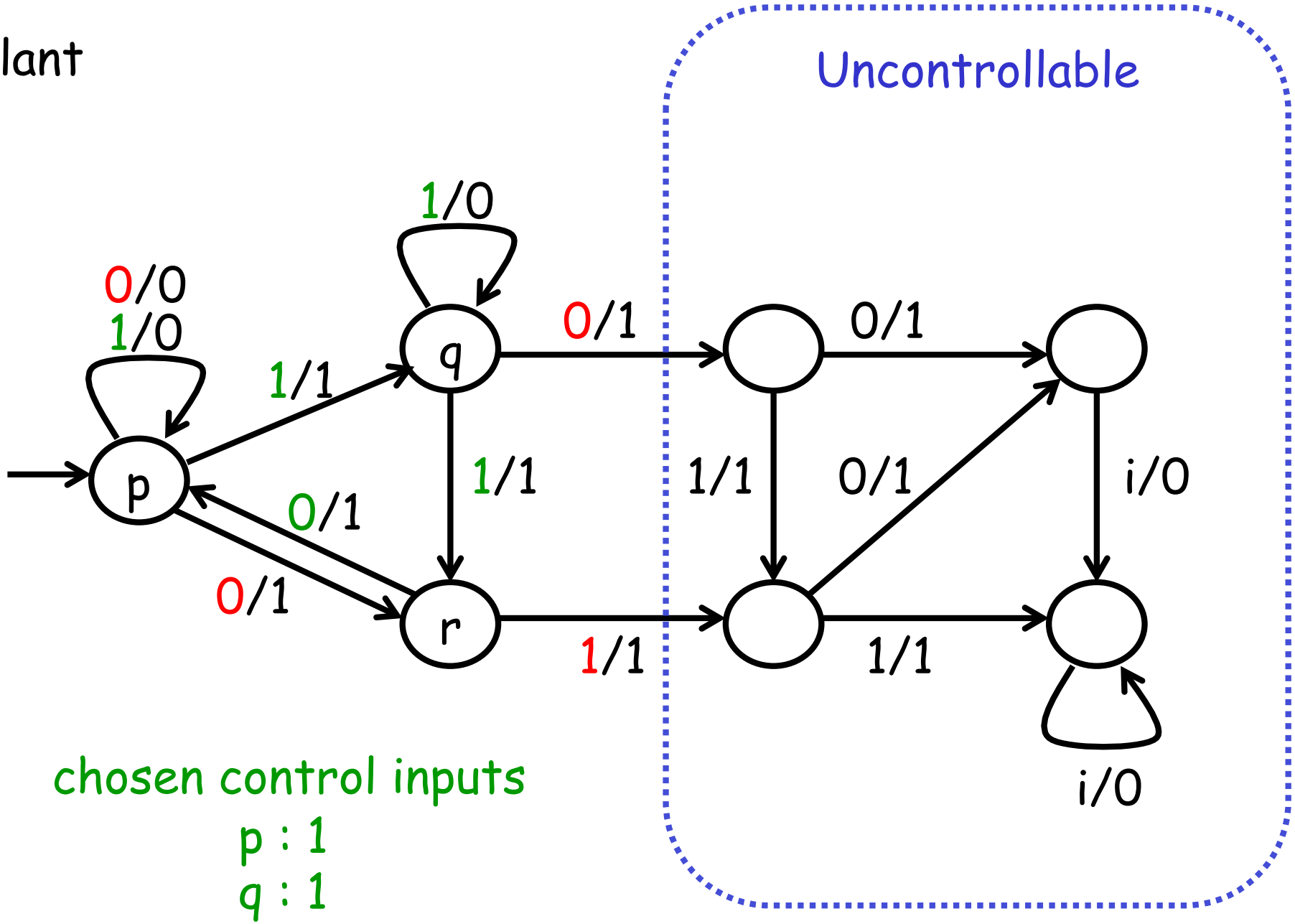

Plant
Uncontrollable
Error
1/0
0/0
1/0
0/1
0/1
1/1
1/1
1/1
0/1
i/0
0/1
0/1
1/1
1/1
i/0
safe control inputs

Step 2:

Design the Controller

1. For each controllable state s of the plant, choose one input i so that possibleUpdates (s,i) contains only controllable states.

Plant



chosen control inputs

p : 1

q : 1

r : 0

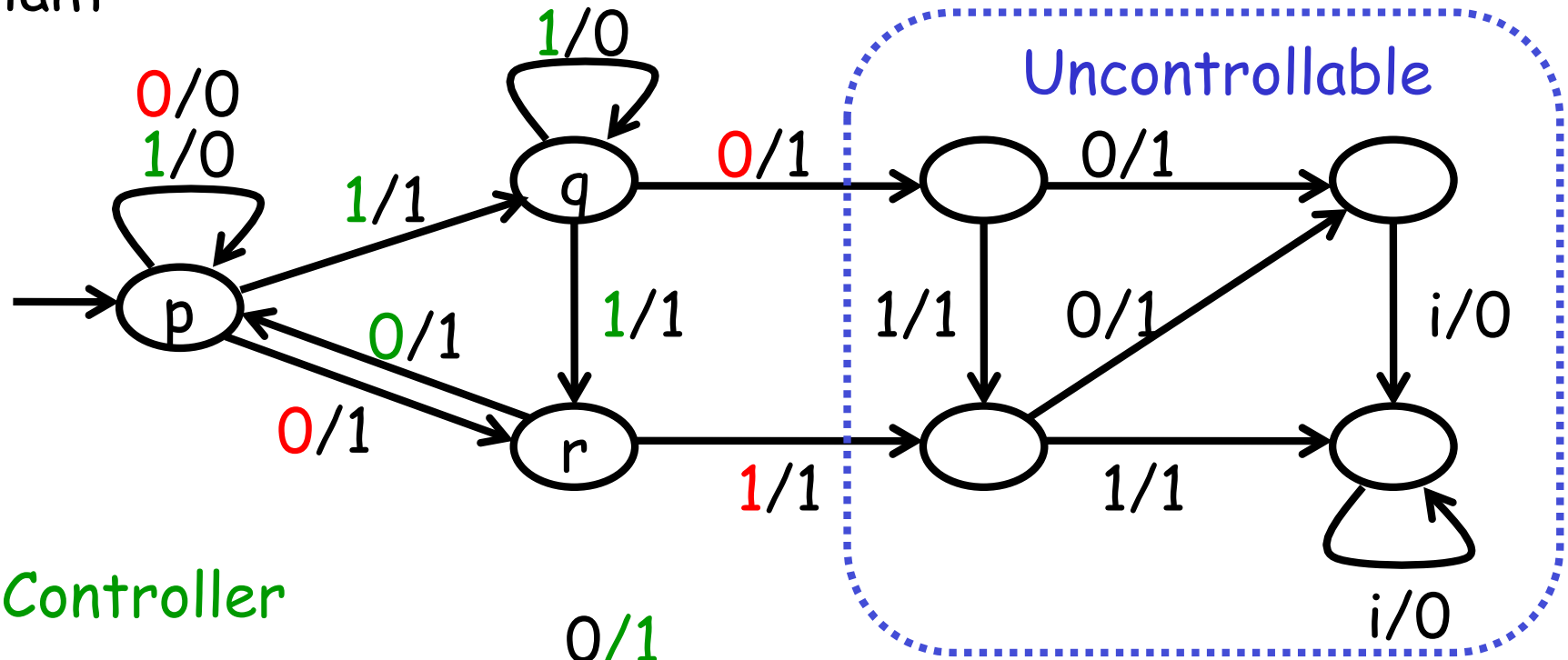
Step 2:

Design the Controller

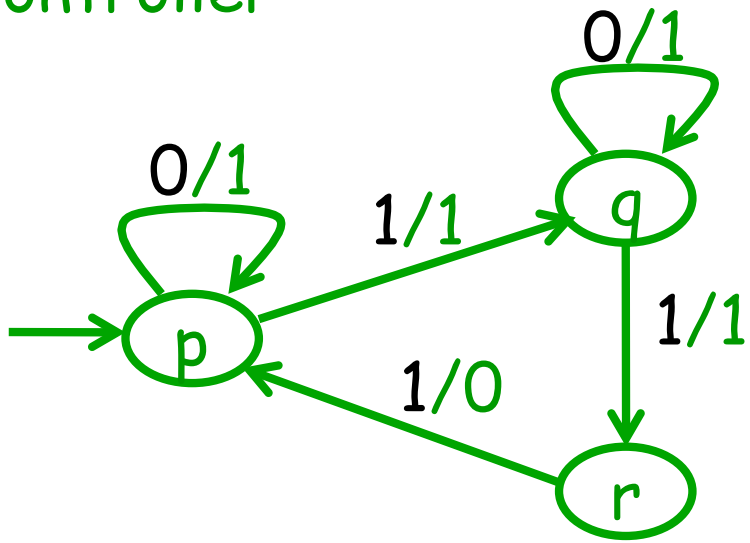
1. For each controllable state s of the plant, choose one input i so that possibleUpdates (s,i) contains only controllable states.
2. Have the Controller keep track of the state of the Plant:

   If Plant is output-deterministic,
   then Controller looks exactly like the controllable part of Plant, with inputs and outputs swapped.

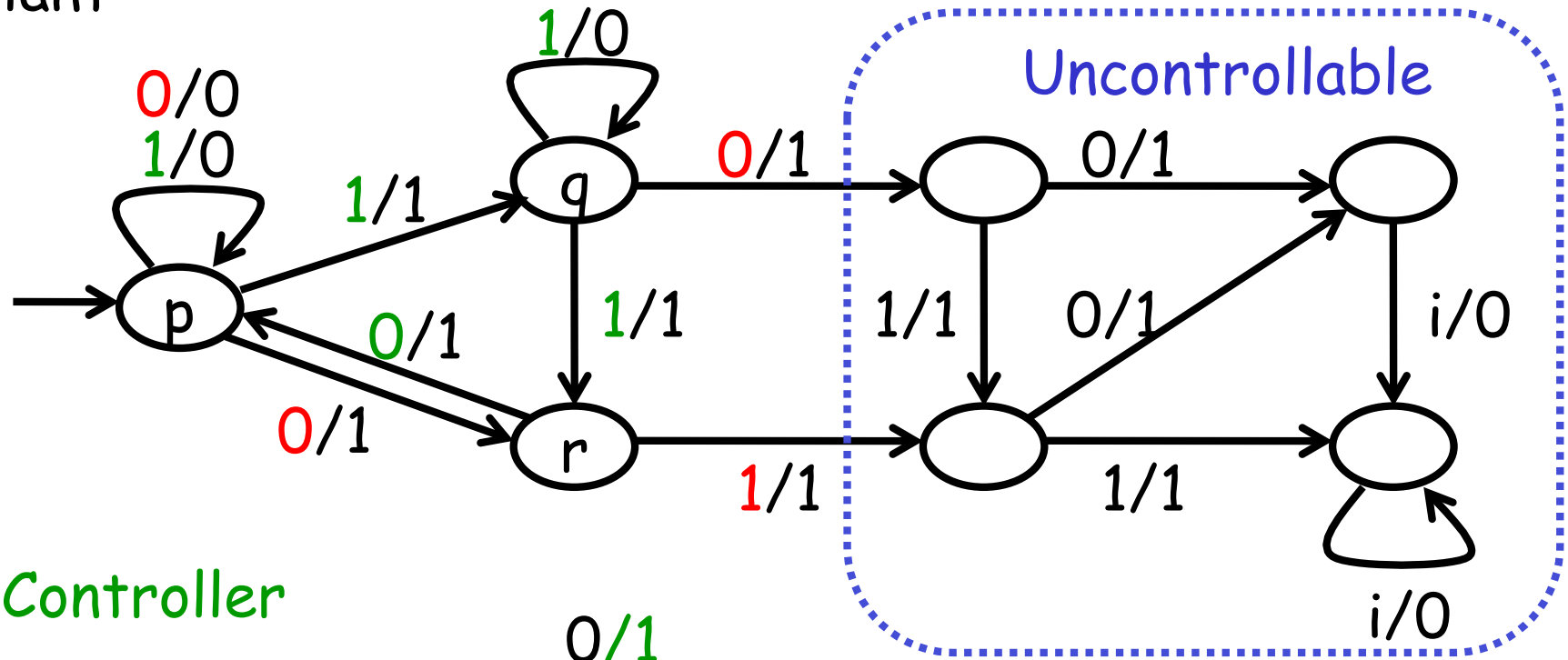
Plant



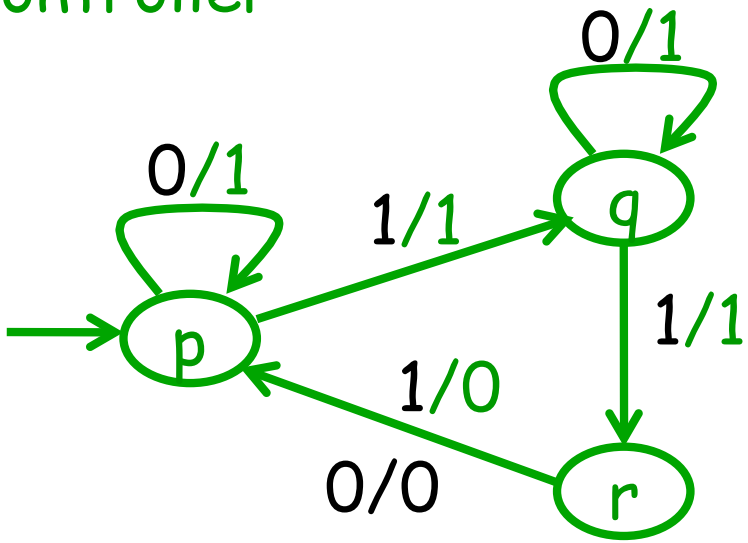
Controller



Plant



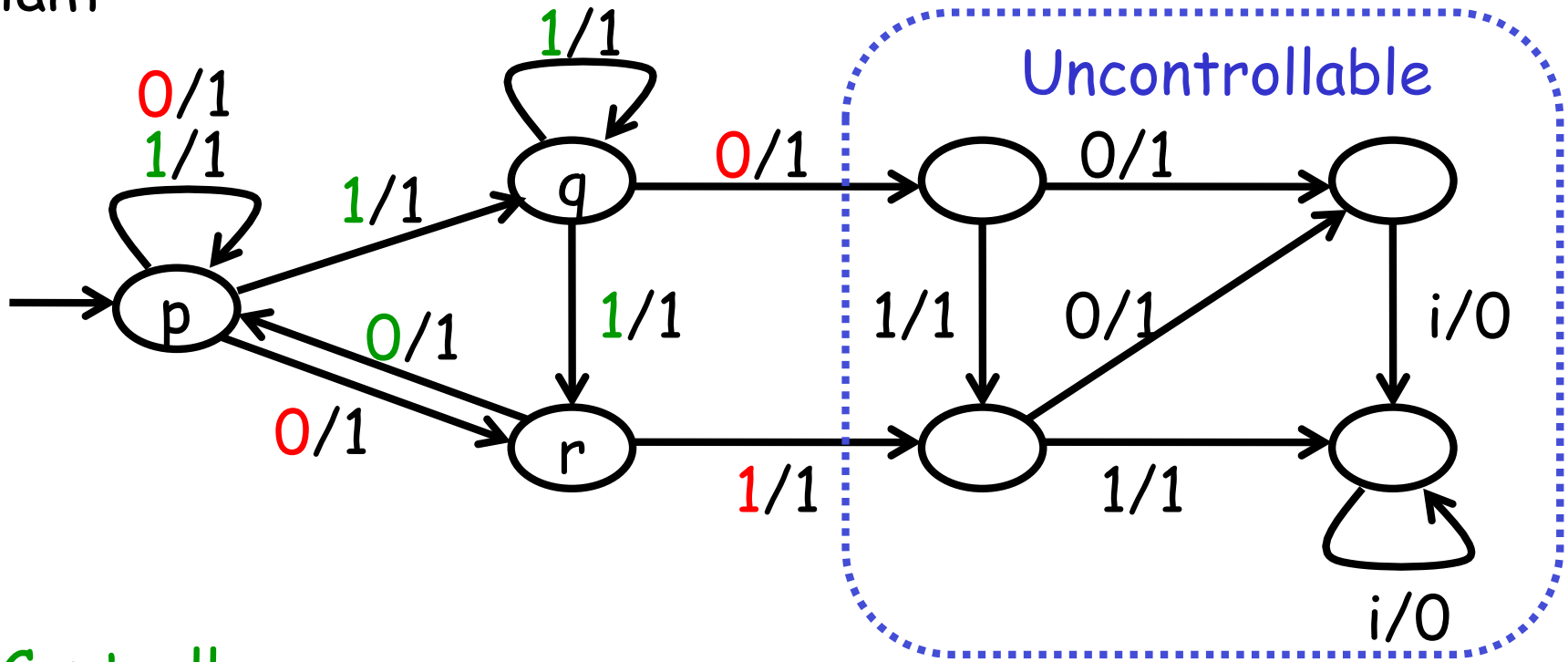
Controller



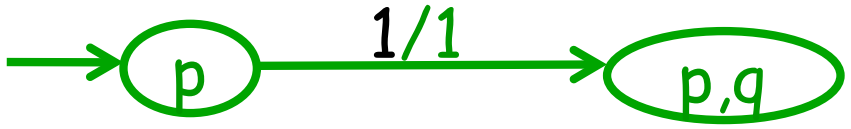
(the Controller can be made receptive in any way)

What if the Plant is not output-deterministic?

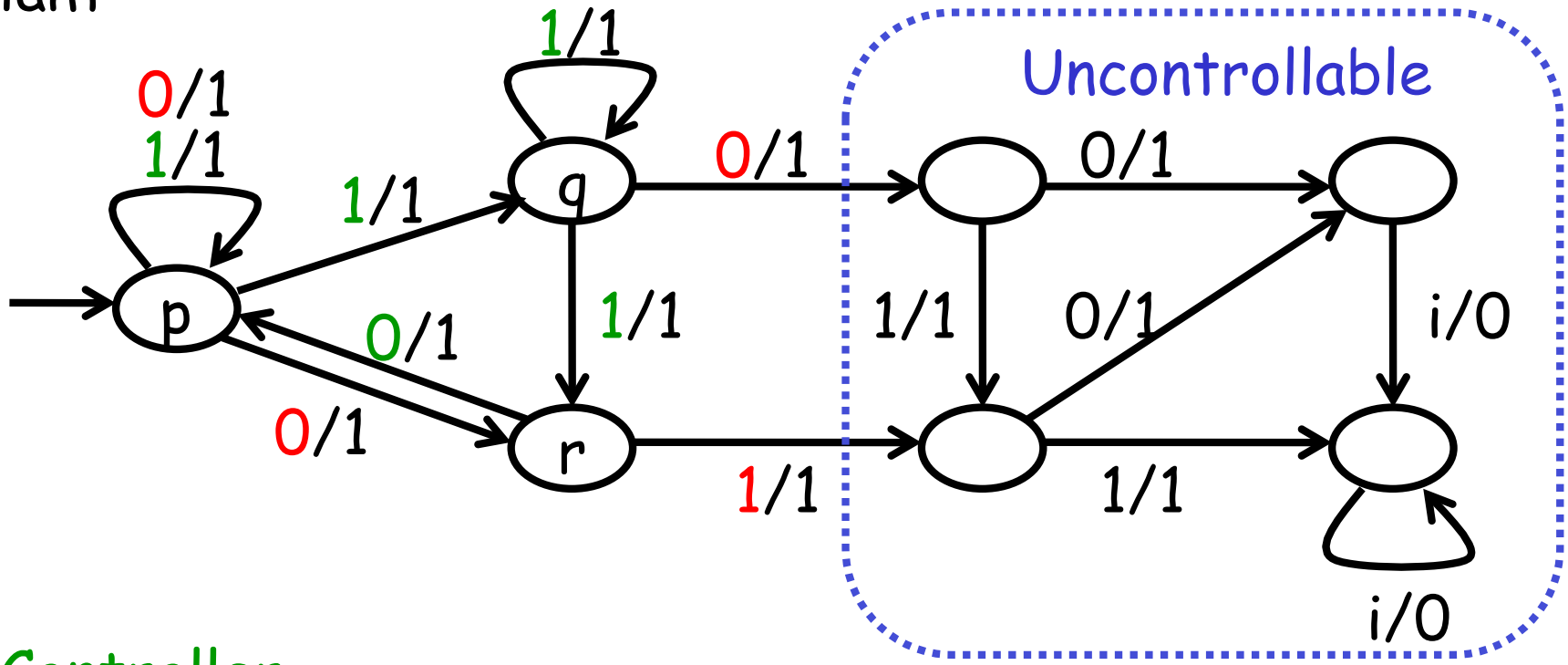
Plant



Controller

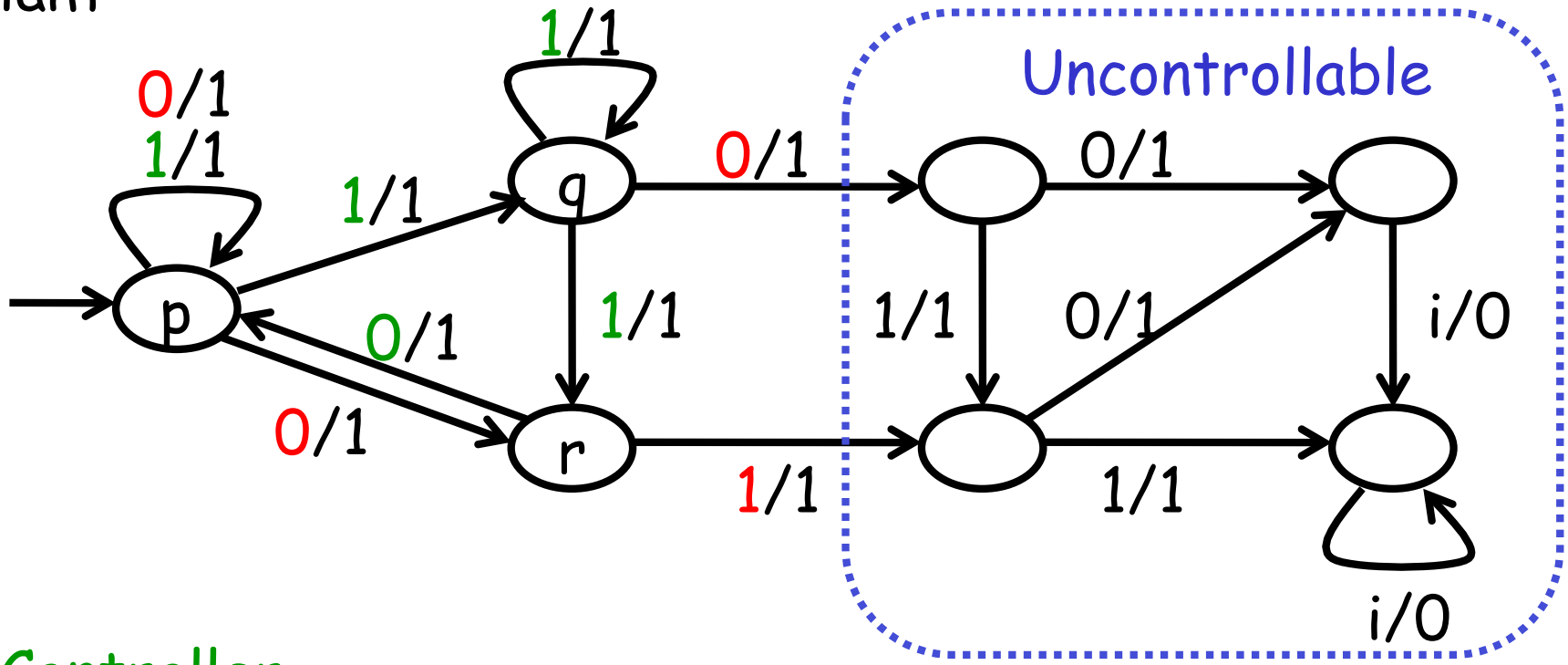


Plant



Controller

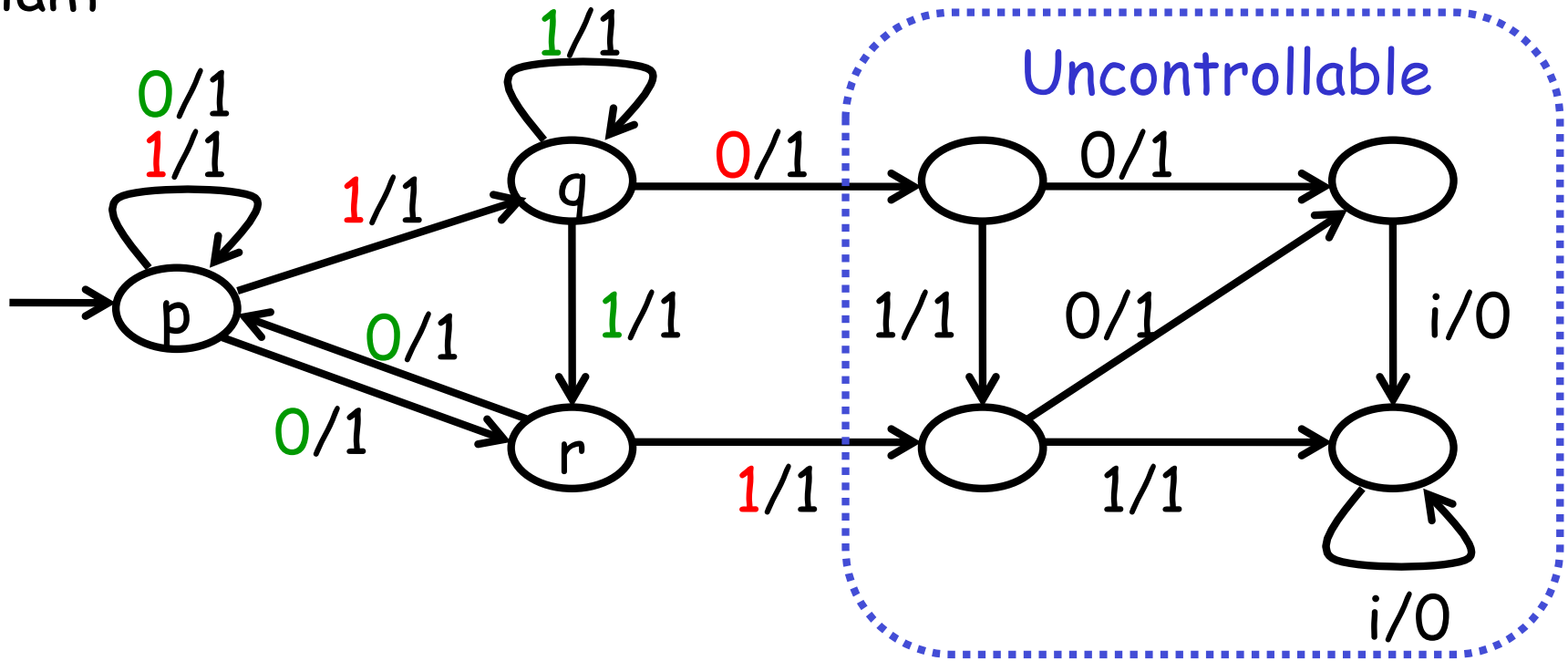
Plant



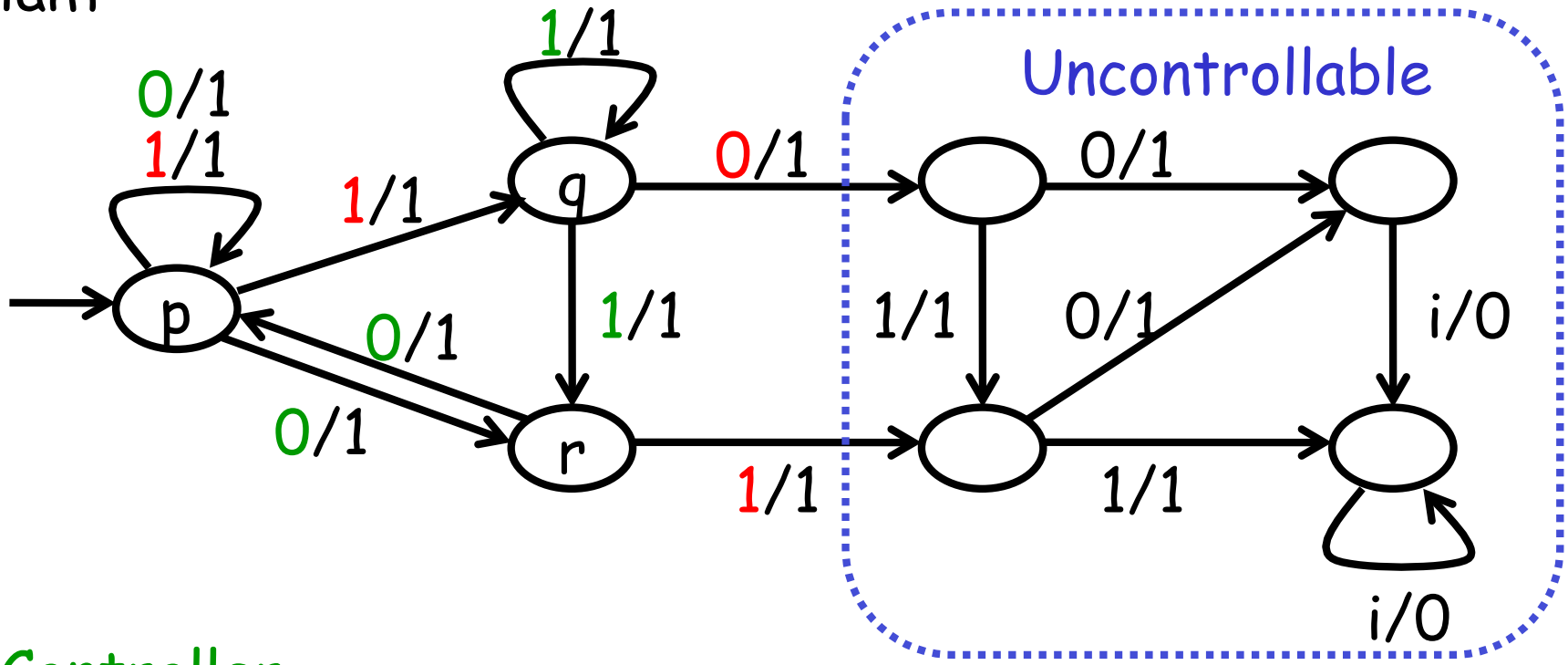
Controller

Neither 0 nor 1 is safe !

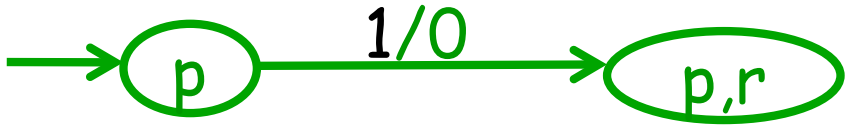
Plant



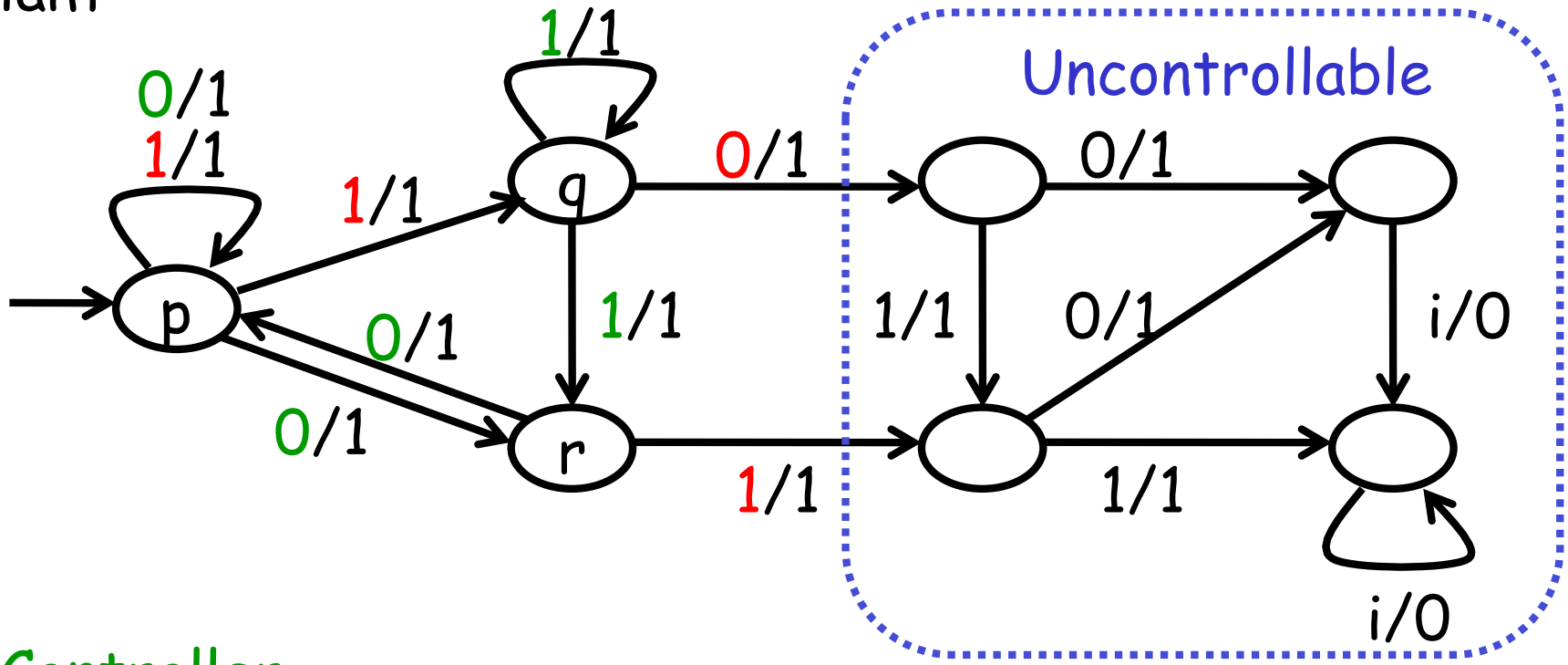
Plant



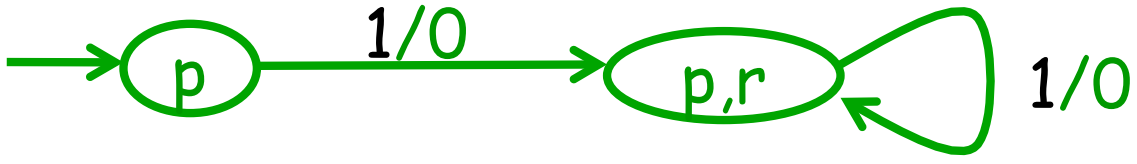
Controller



Plant



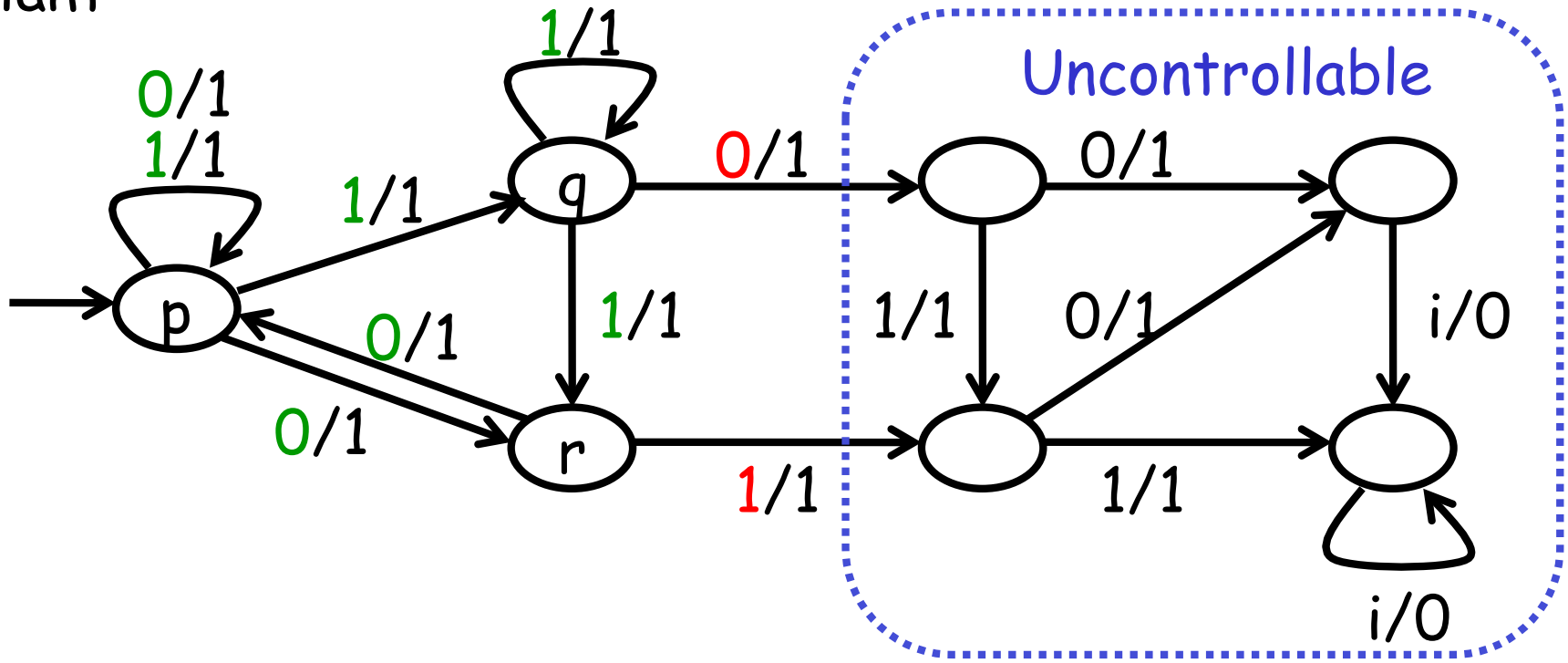
Controller



Step 2: Design the Controller

1. Let Controllable be the controllable states of the Plant. A subset $S \subseteq$ Controllable is consistent if there is an input i such that for all states $s \in S$, all states in possibleUpdates (s,i) are controllable.
2. Let M be the state machine whose states are the consistent subsets of Controllable. Prune from M the states that have no successor, until no more states can be pruned.
3. If the result contains possibleInitialStates (of the plant) as a state, then it is the desired Controller. Otherwise, no controller exists.

Plant

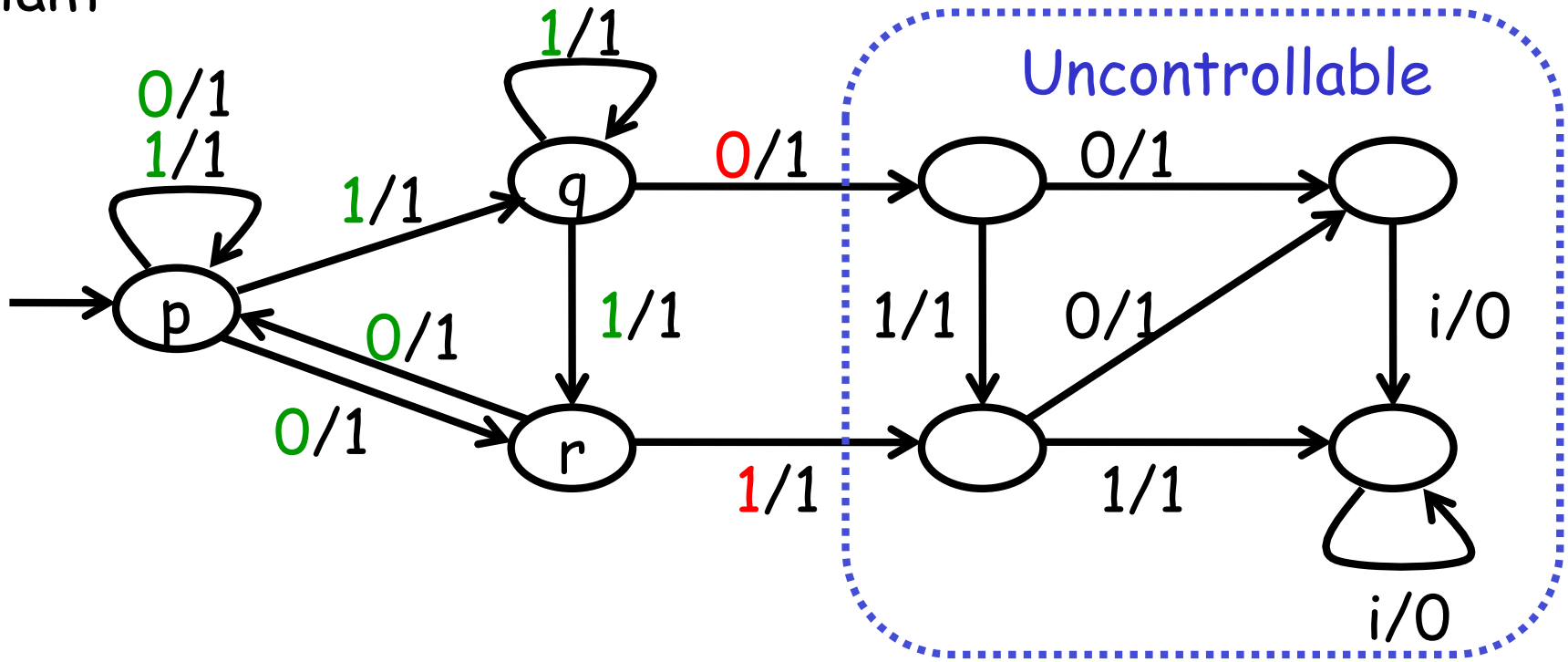


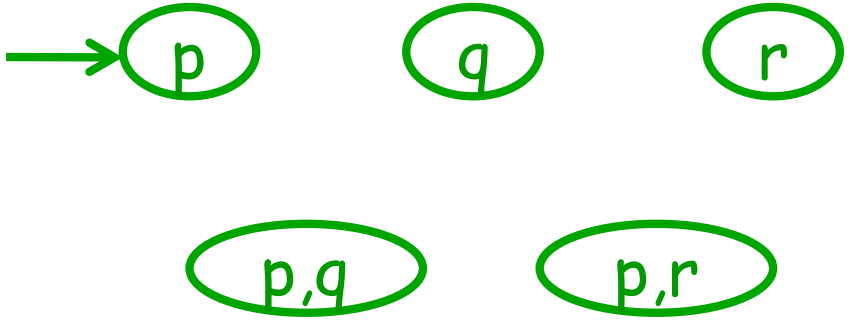
Consistent subsets

{p} : 0, 1
{q} : 1
{r} : 0

{p,q} : 1
{p,r} : 0
{q,r}, {p,q,r} not consistent

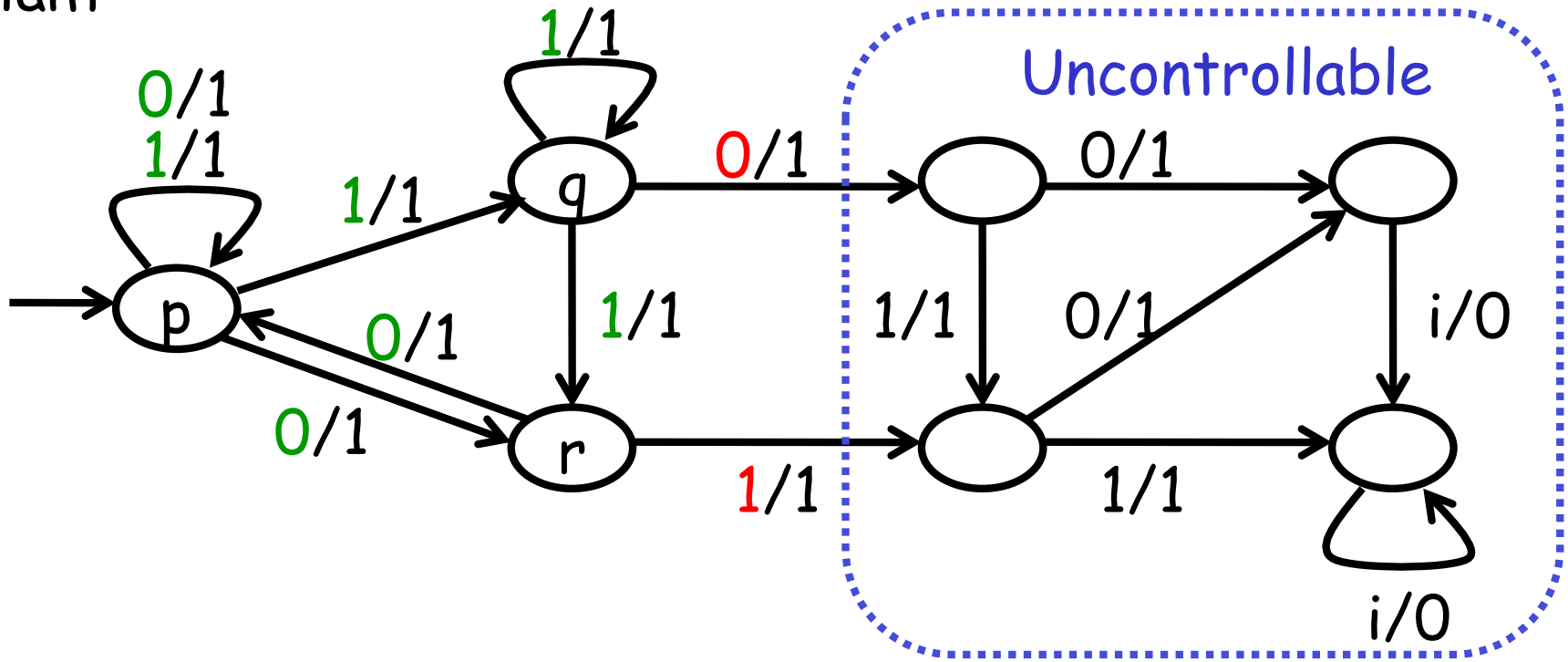
Plant

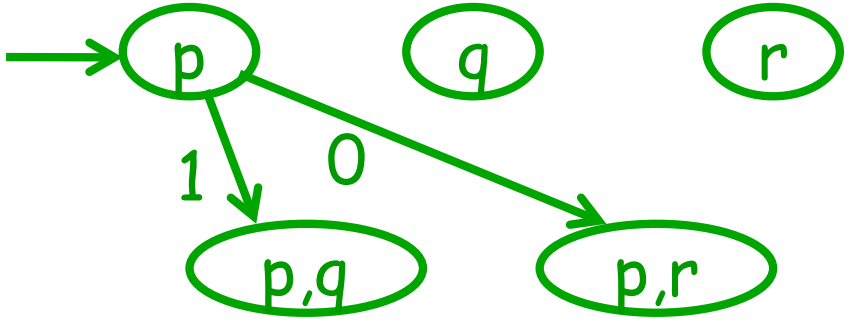




{p} : 0, 1
{q} : 1
{r} : 0
{p,q} : 1
{p,r} : 0

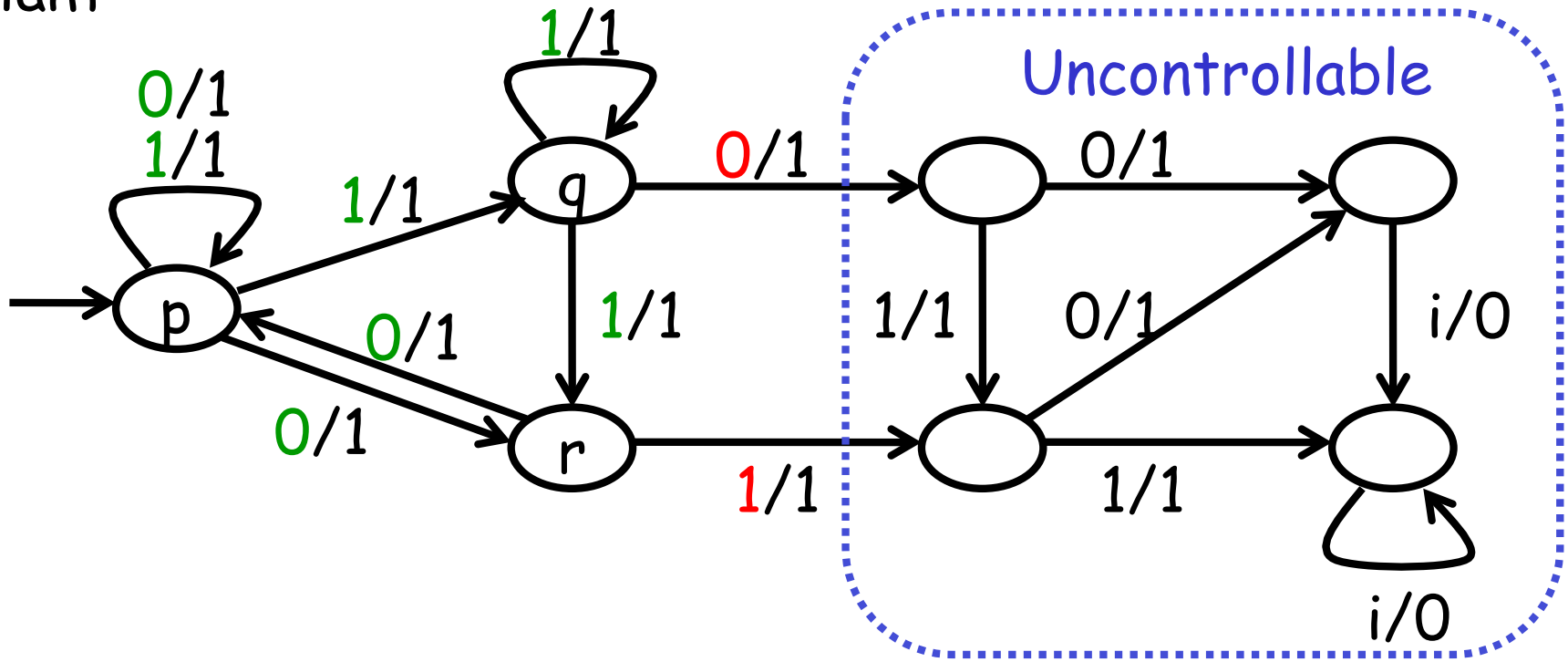
Plant

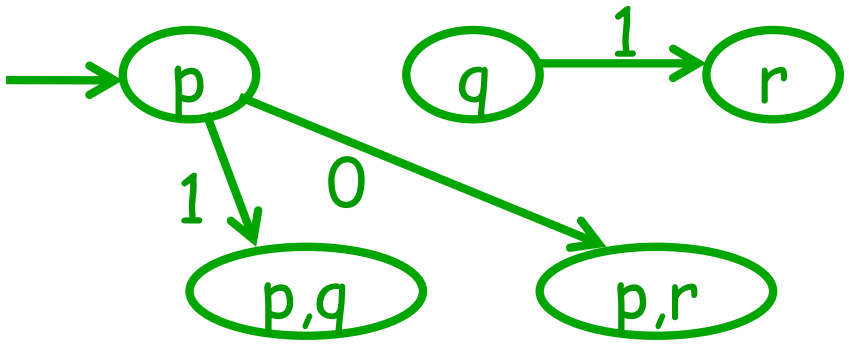




{q} : 1
{r} : 0
{p,q} : 1
{p,r} : 0

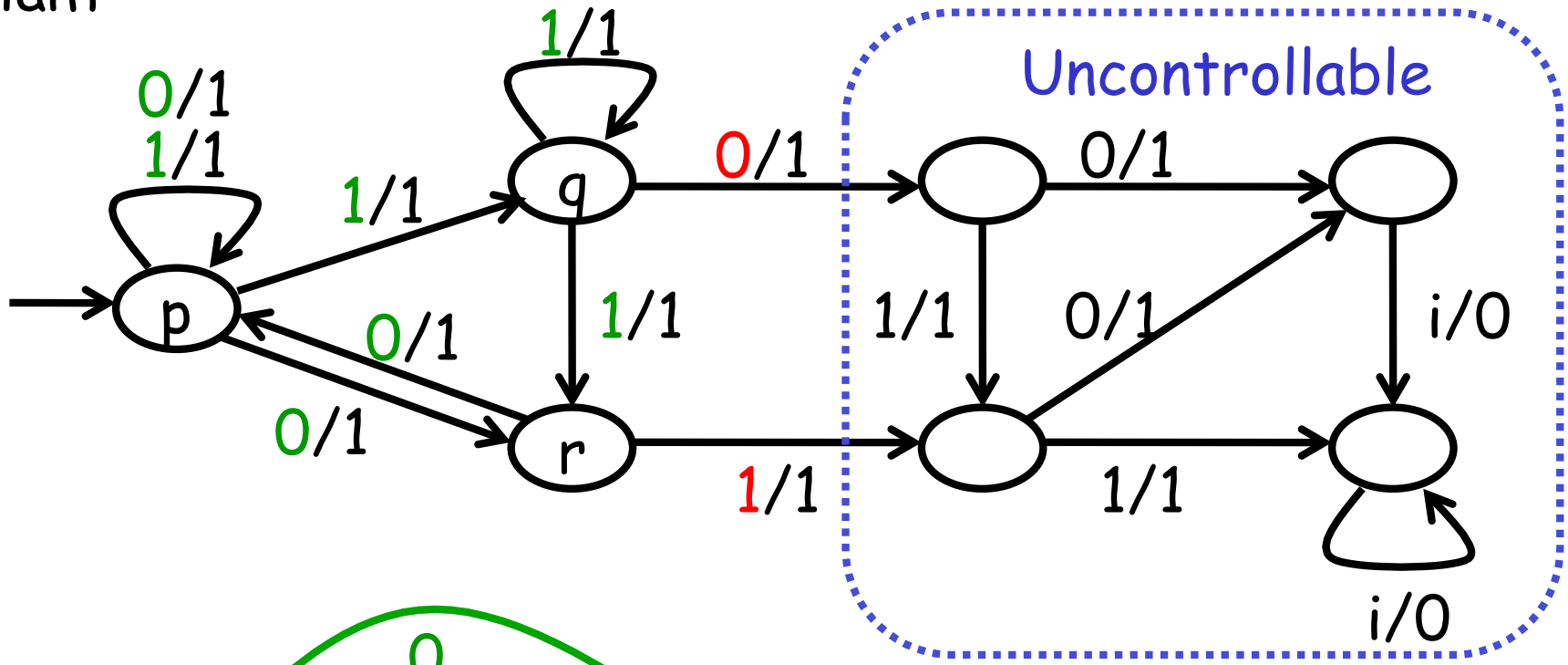
Plant

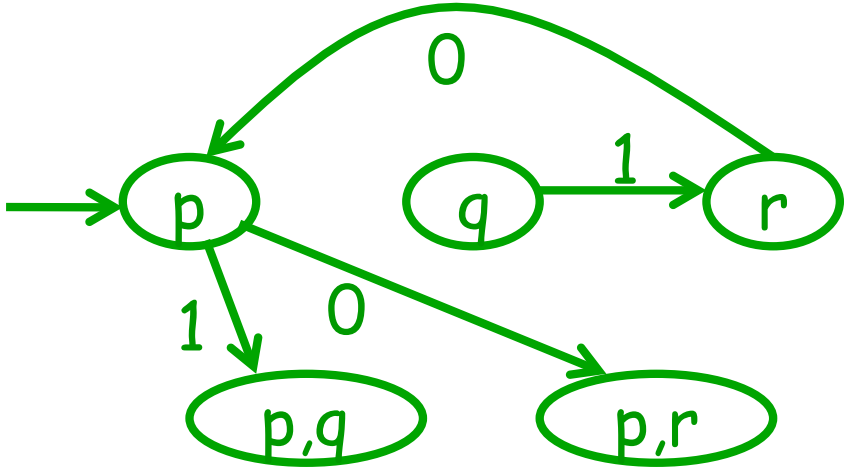




{r} : 0
{p,q} : 1
{p,r} : 0

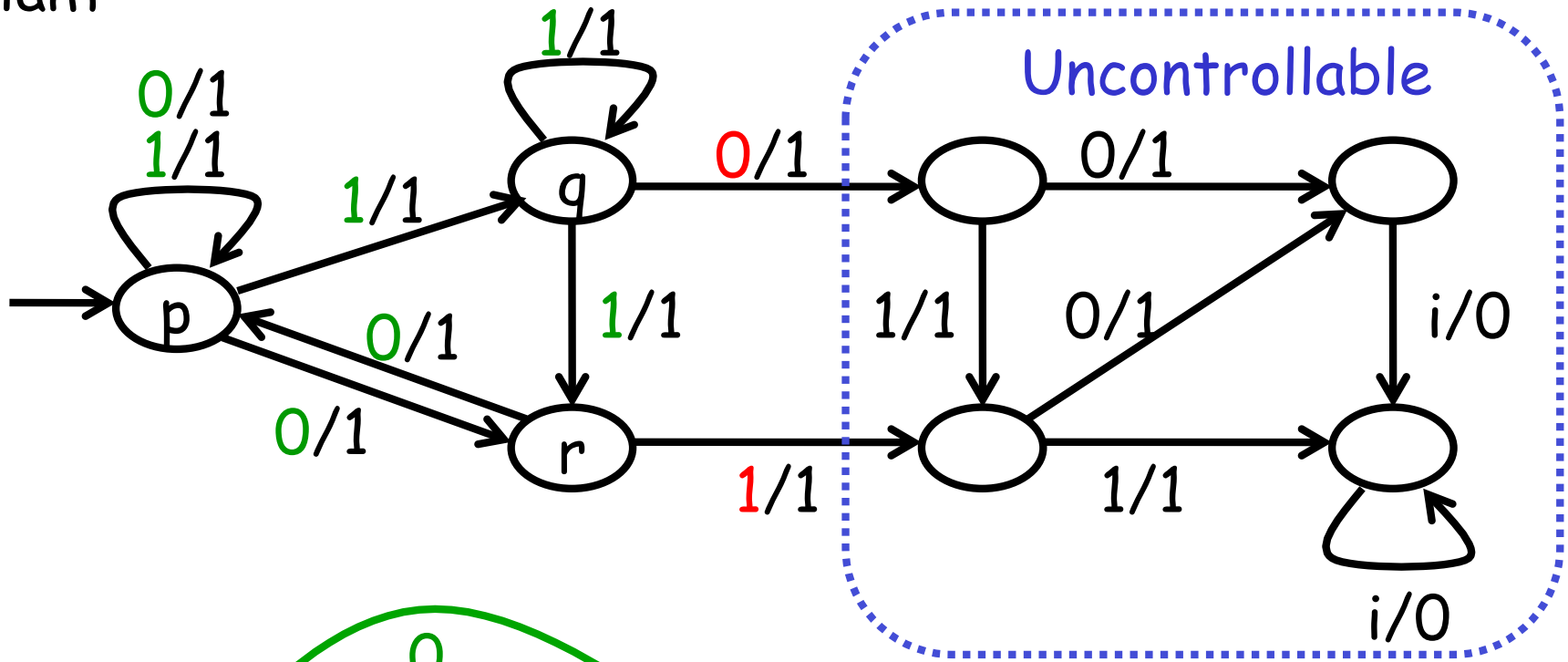
Plant

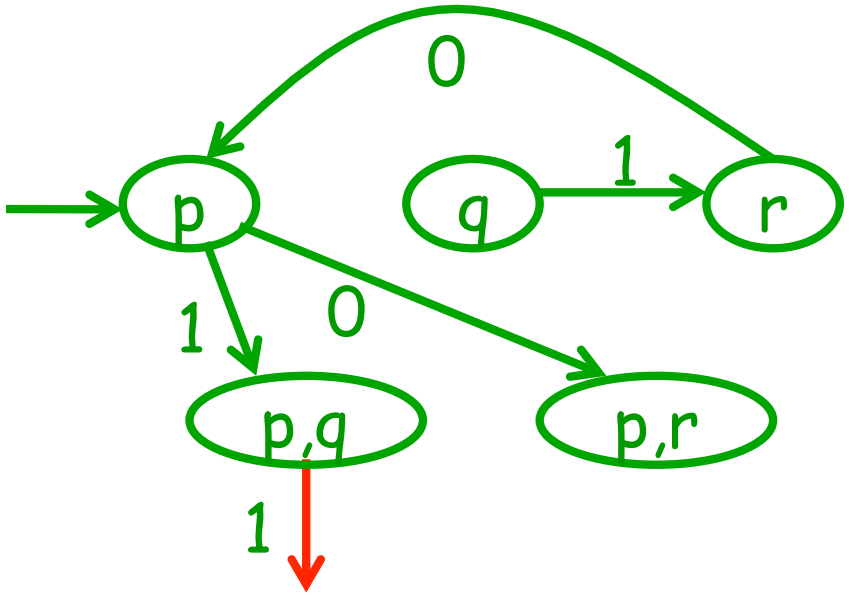




{p,q} : 1
{p,r} : 0

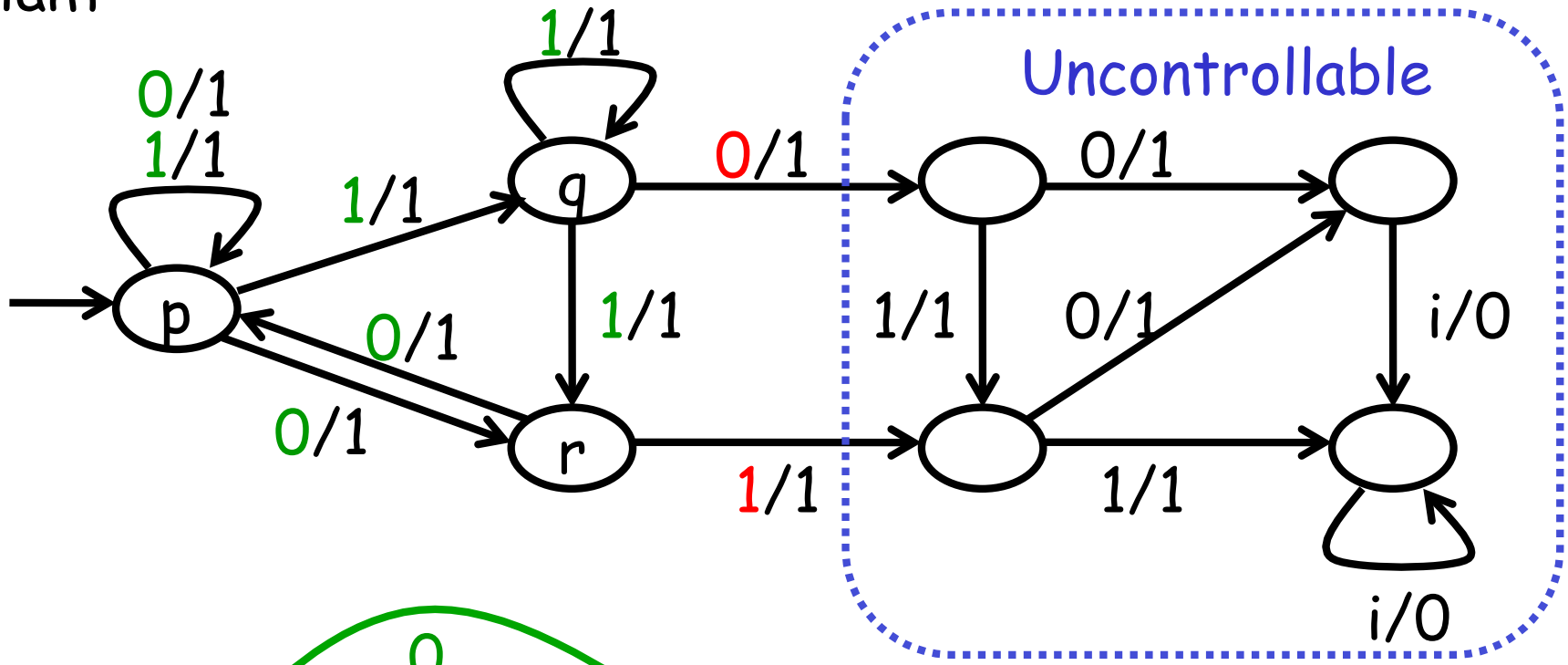
Plant

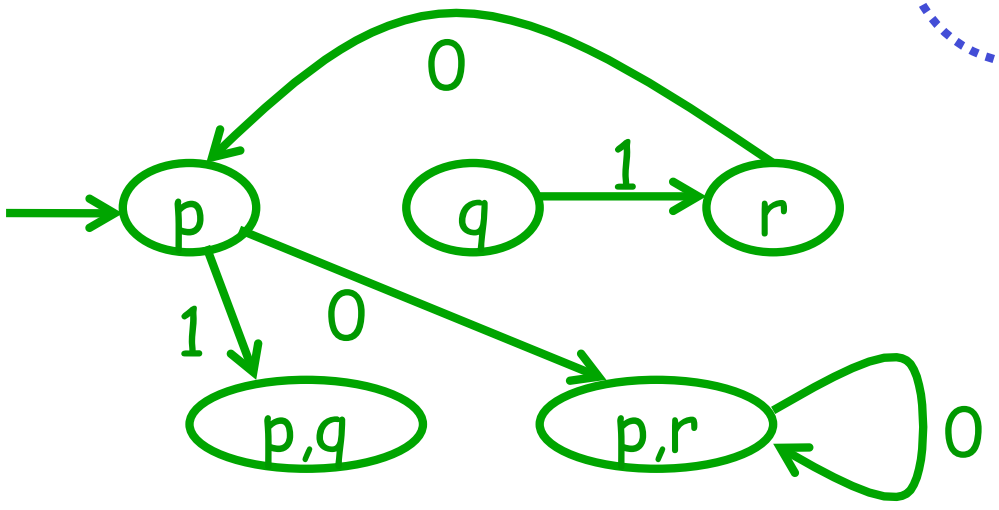

Uncontrollable
0/1
1/1
1/1
q
0/1
0/1
1/1
p
1/1
0/1
0/1
r
1/1
1/1
0/1
i/0
i/0


0
p
q
1
r
1
0
p,q
p,r
1

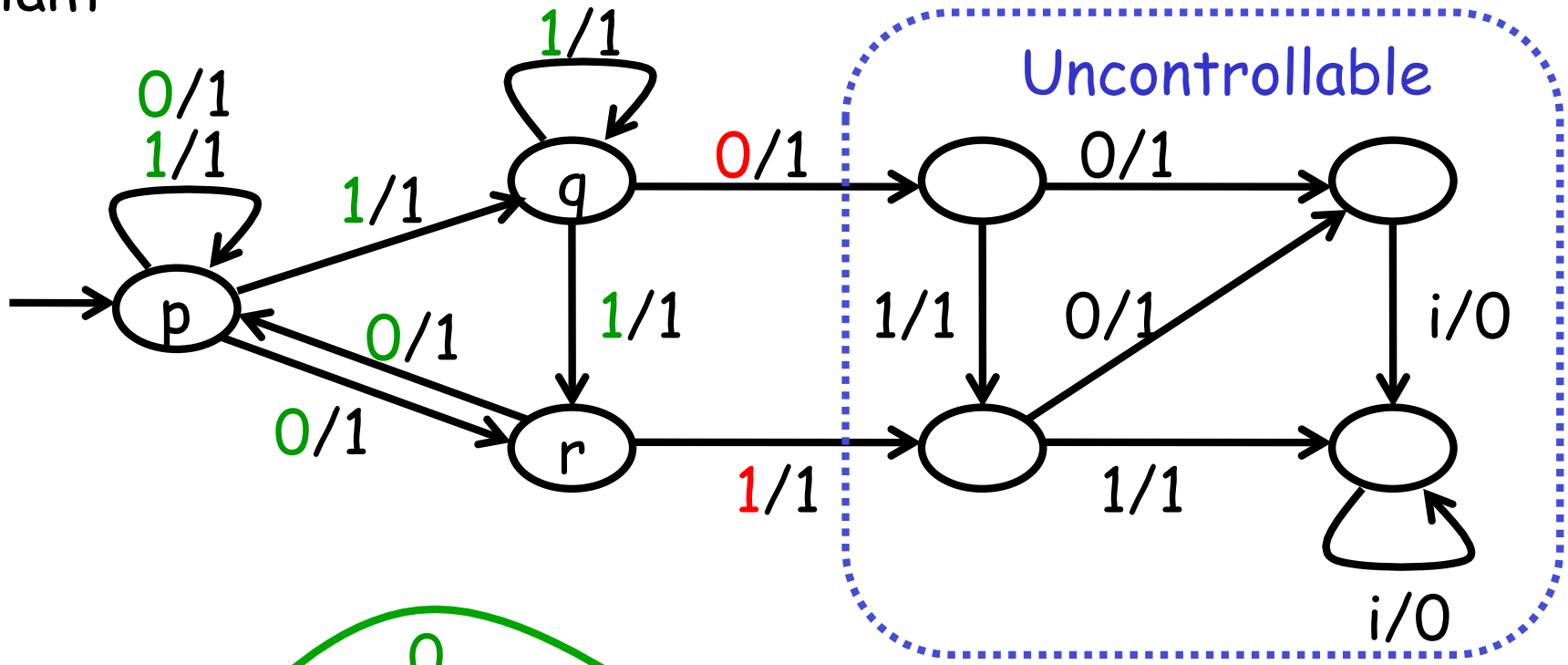
{p,r} : 0

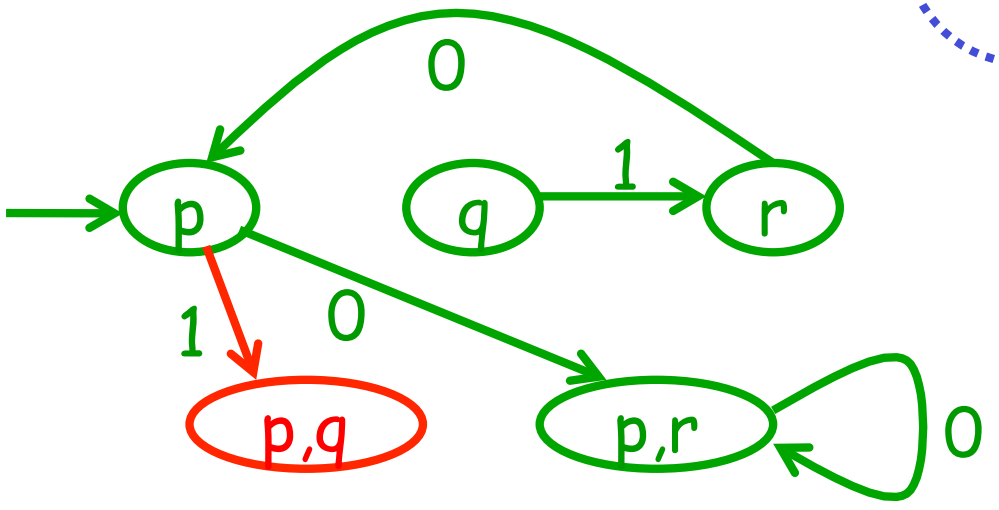
Plant


Uncontrollable
0/1
1/1
1/1
q
0/1
0/1
1/1
p
1/1
0/1
0/1
r
1/1
1/1
0/1
i/0
i/0

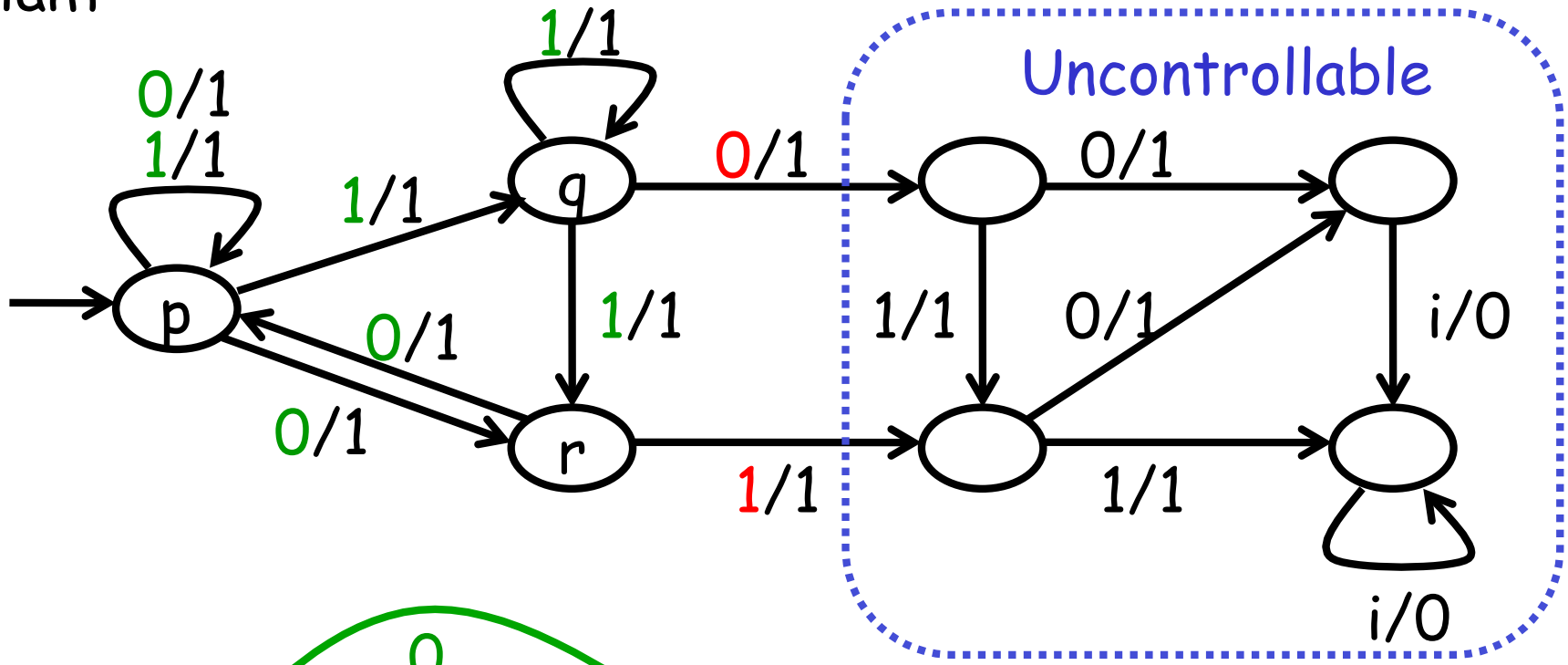

0
p
q
1
r
1
0
p,q
p,r
0

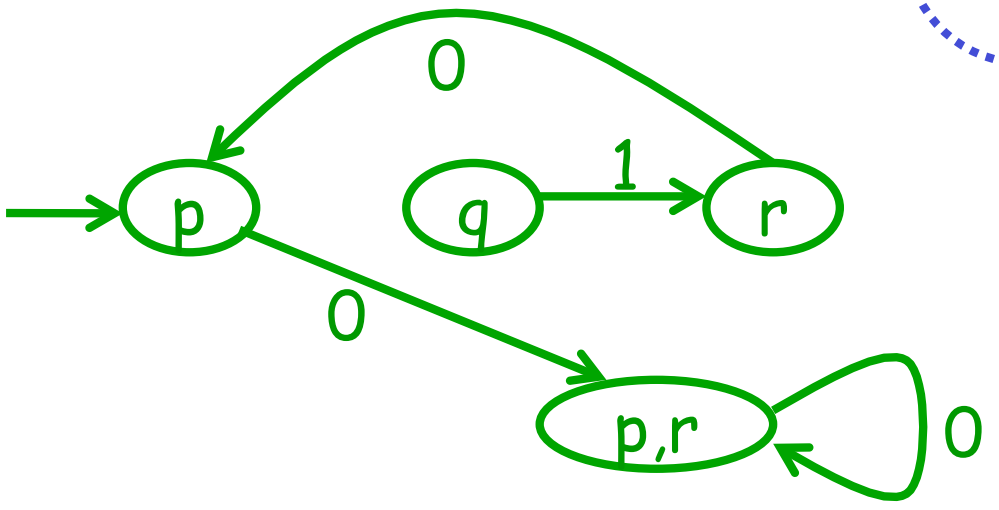
Plant


Uncontrollable
0/1
1/1
1/1
q
0/1
0/1
1/1
p
1/1
0/1
0/1
r
1/1
1/1
0/1
i/0
i/0

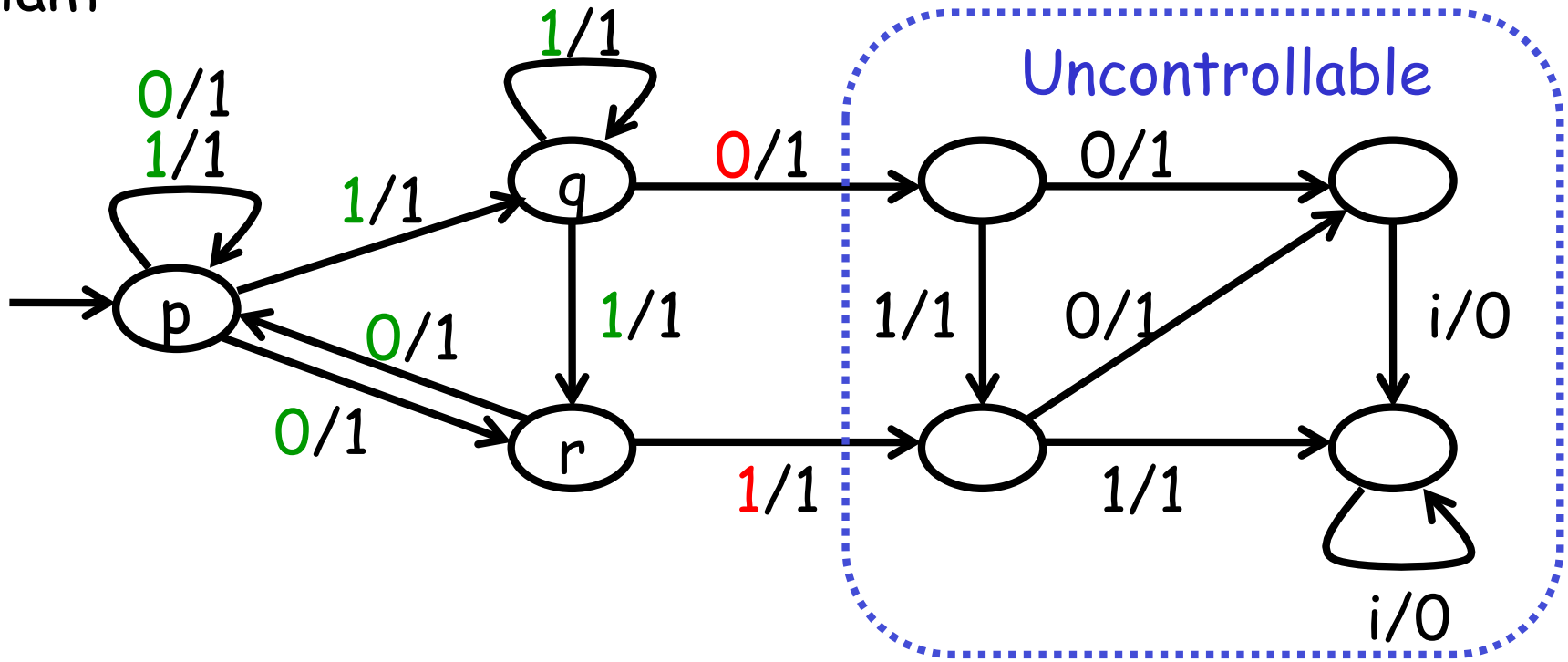

0
p
q
1
r
1
0
p,q
p,r
0

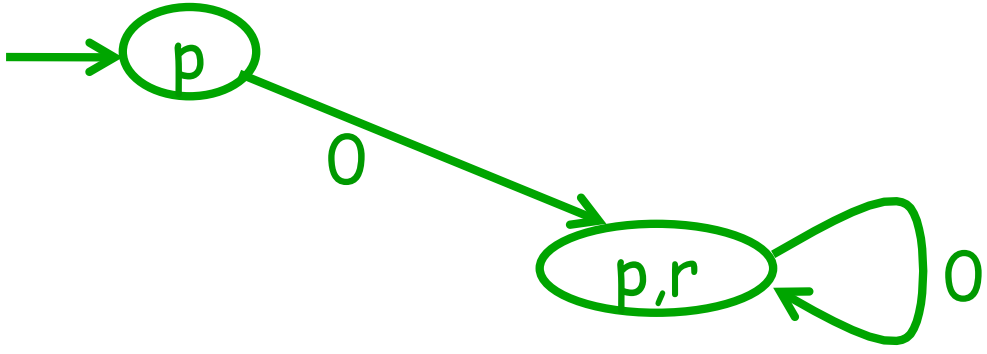
Plant


Uncontrollable
0/1
1/1
1/1
q
0/1
0/1
1/1
p
1/1
0/1
0/1
r
1/1
1/1
0/1
i/0
i/0

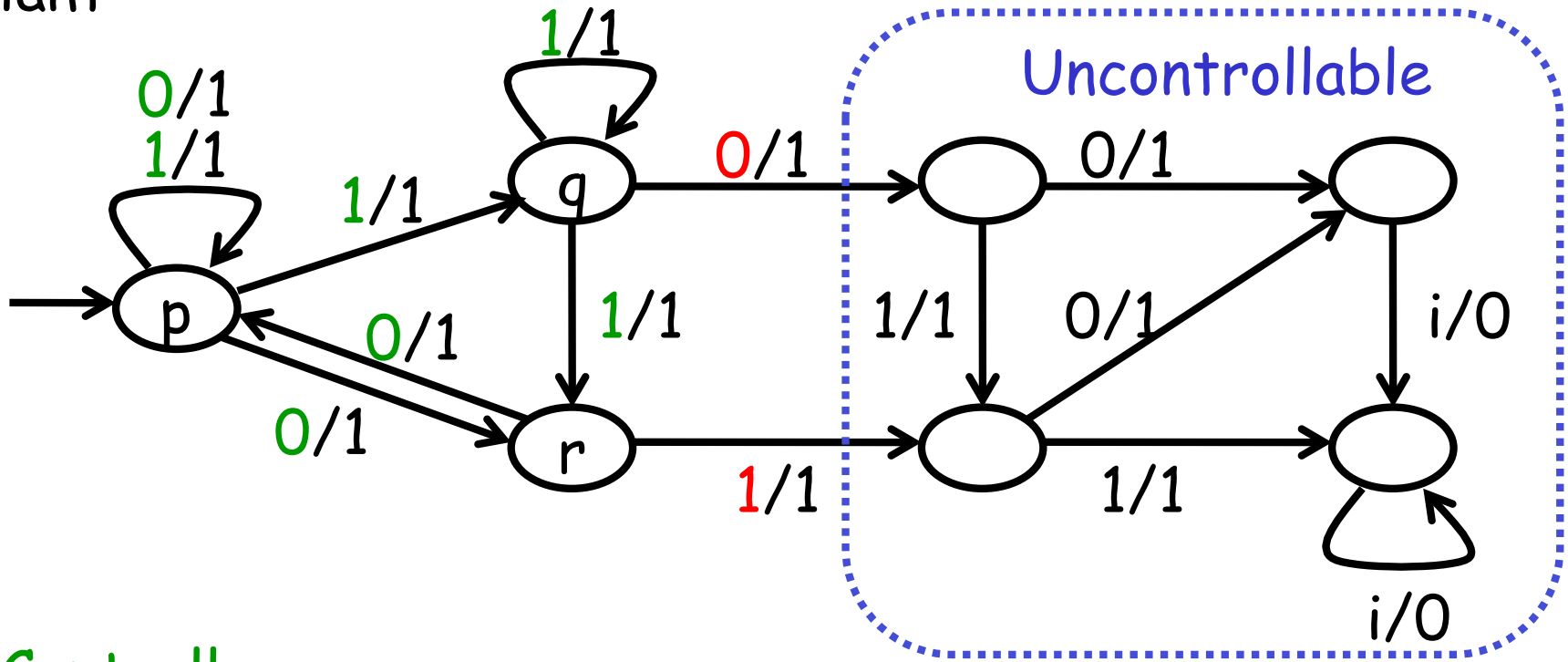

0
p
q
1
r
0
p,r
0

Plant

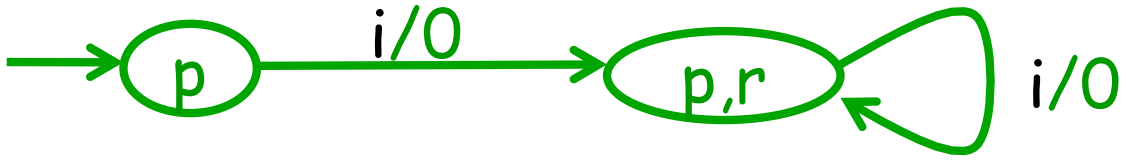

Uncontrollable
0/1
1/1
1/1
q
0/1
0/1
1/1
p
0/1
1/1
0/1
i/0
0/1
r
1/1
1/1
i/0


p
0
p,r
0

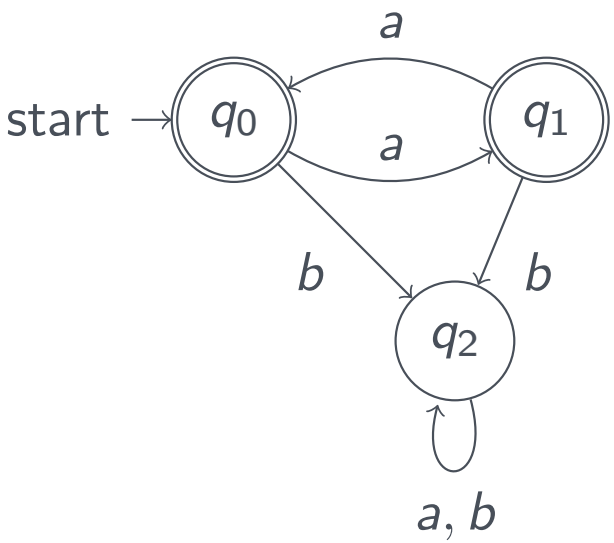
Plant

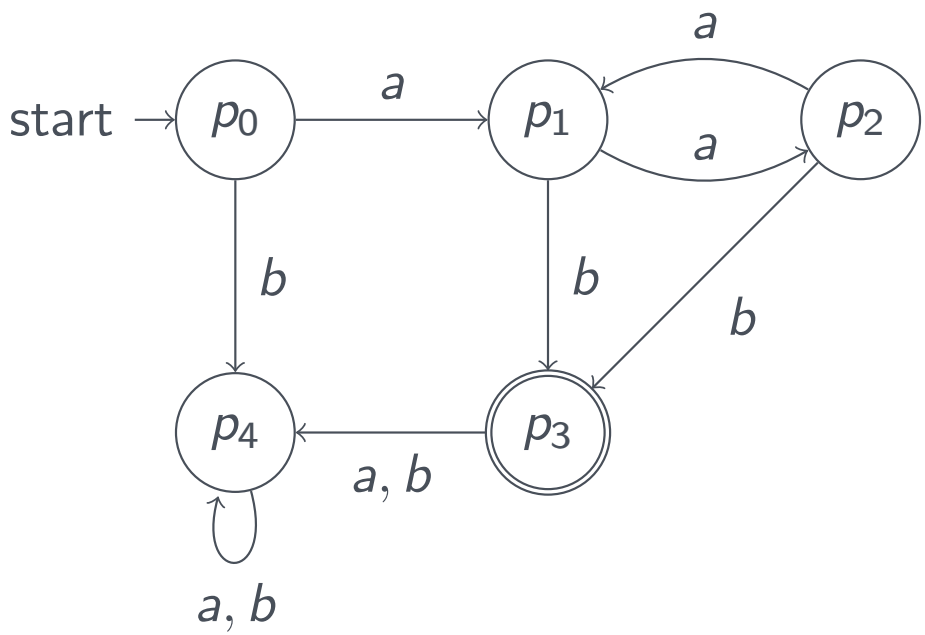


Controller



1. Costruire l'automa che riconosce l'intersezione dei linguaggi dei seguenti due automi:





Esercizi

Per ognuno dei seguenti linguaggi, costruire un DFA sull'alfabeto $\{0, 1\}$ che li rappresenti:

2. Tutte le stringhe w che contengono la sottostringa 101
3. Tutte le stringhe w che non contengono la sottostringa 101

Esercizi

4. Sia L un linguaggio sull'alfabeto Σ e $a \in \Sigma$. Definiamo il quoziente di L e a come il linguaggio:

$$L/a = \{w \in \Sigma^* : wa \in L\}$$

   Dimostrare che se L è regolare allora anche L/a è regolare.

5. Sia L un linguaggio regolare. Dimostrare che anche i seguenti linguaggi sono regolari:
   - $\min(L) = \{w : w \in L$ ma nessun prefisso proprio di w è in $L\}$
   - $\max(L) = \{w : w \in L$ e per nessuna stringa $x \neq \varepsilon$, $wx \in L\}$