Hyper-Minimization in $O(n^2)$

Andrew Badr
andrewbadr@gmail.com
CIAA 2008

What is hyper-minimization?

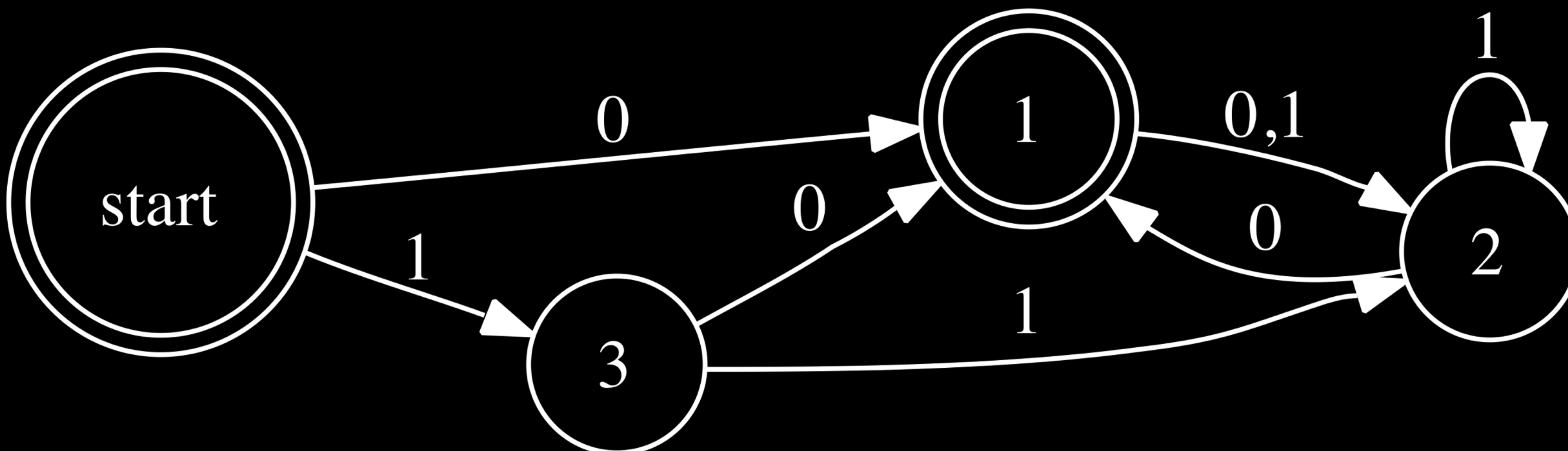
Classical Equivalence

- D_1 and D_2 recognize the same language.
- $L(D_1) \otimes L(D_2)$ is empty
- *Notation:* $D_1 \approx D_2$

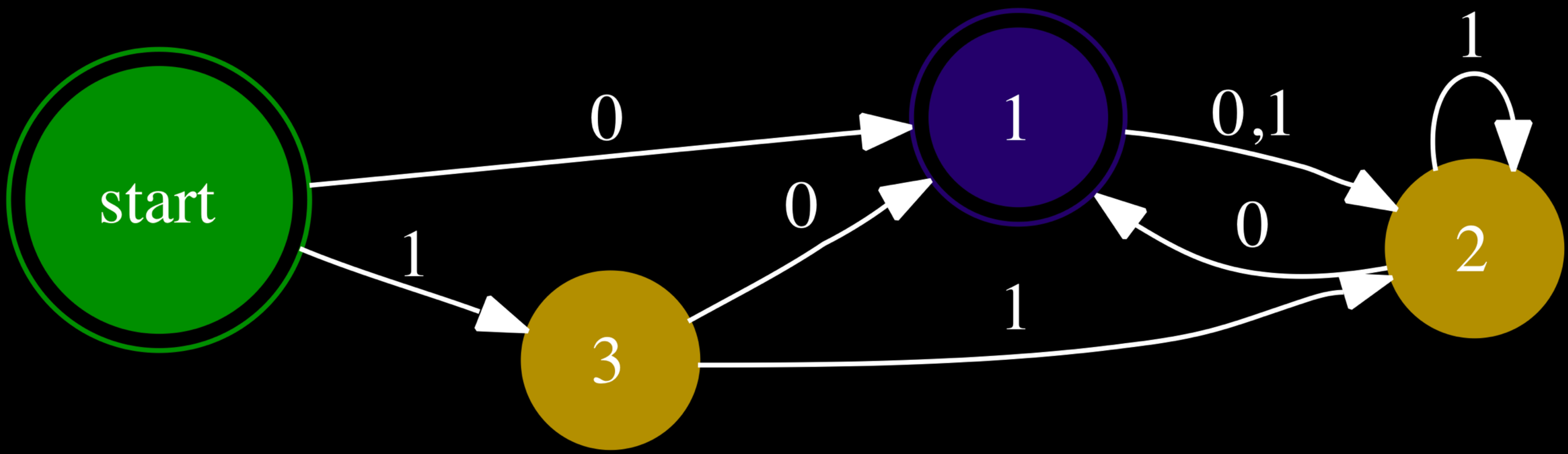
What is hyper-minimization?

Classical Equivalence	F-Equivalence
• D_1 and D_2 recognize the same language. • $L(D_1) \otimes L(D_2)$ is empty • *Notation:* $D_1 \approx D_2$	• D_1 and D_2 "almost" recognize the same language. • $L(D_1) \otimes L(D_2)$ is finite • *Notation:* $D_1 \sim D_2$

Small Example

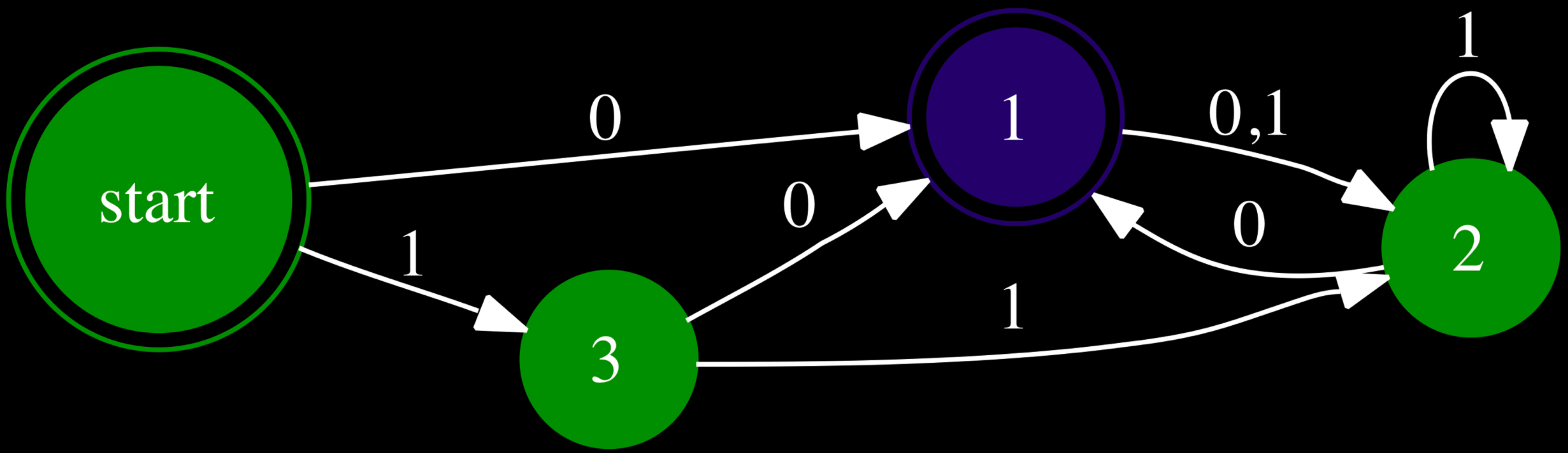


Small Example



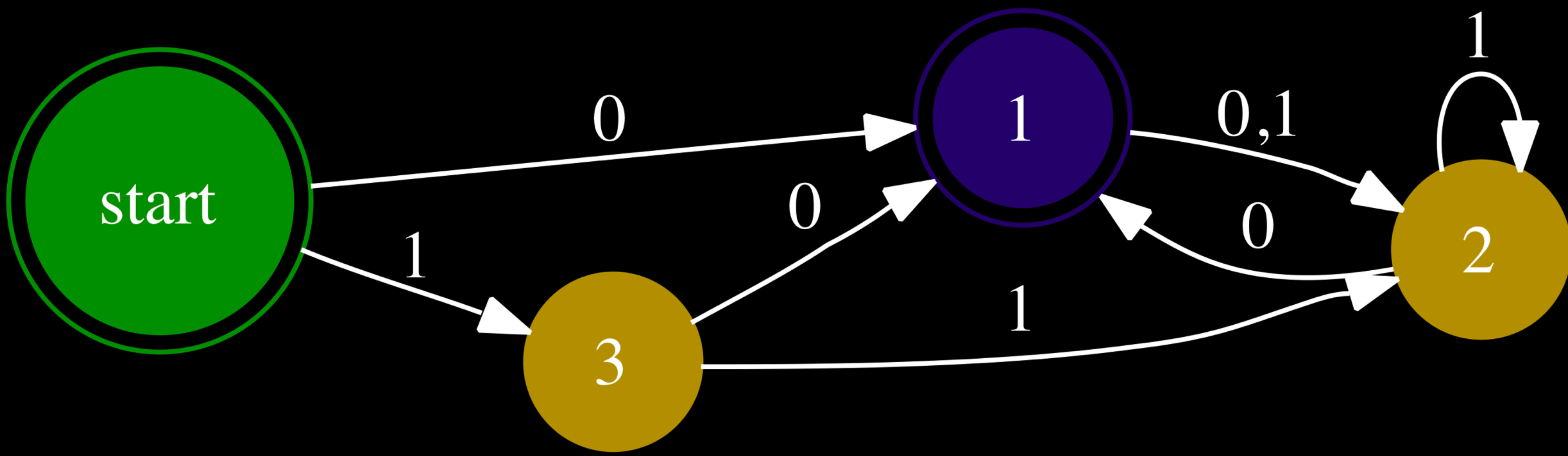
Showing Myhill-Nerode equivalence classes

Small Example



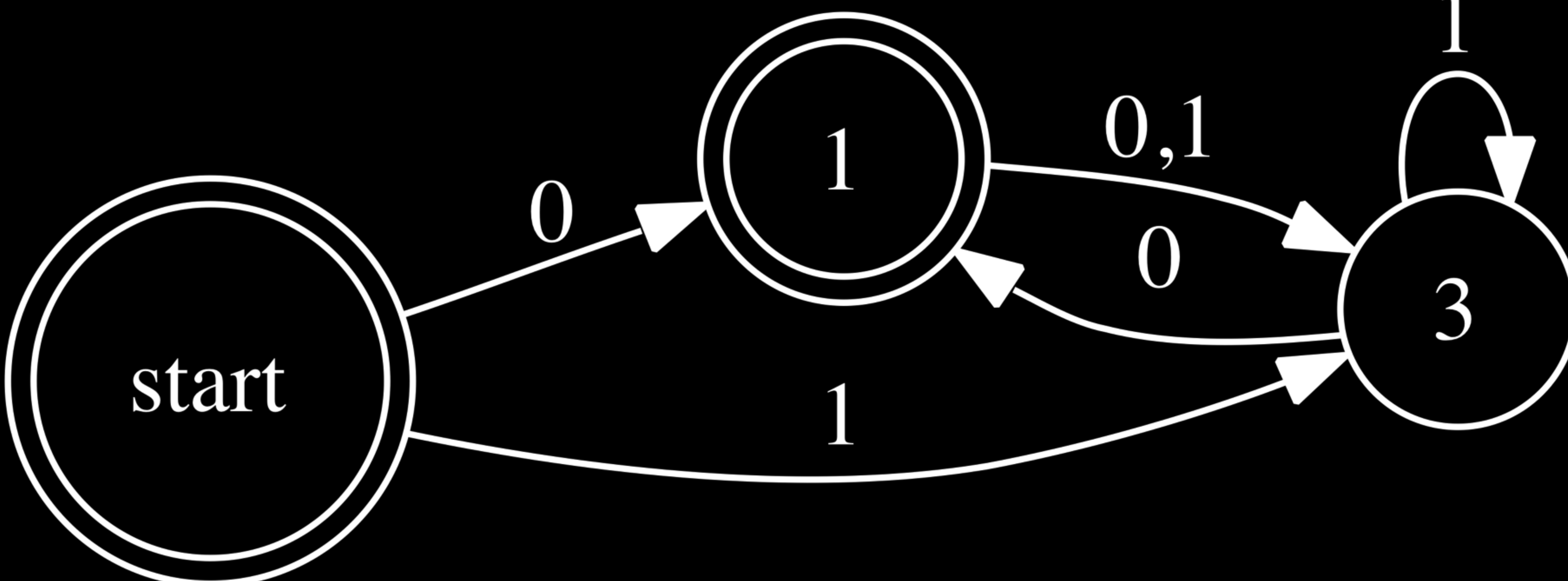
Showing f-equivalence classes

Small Example

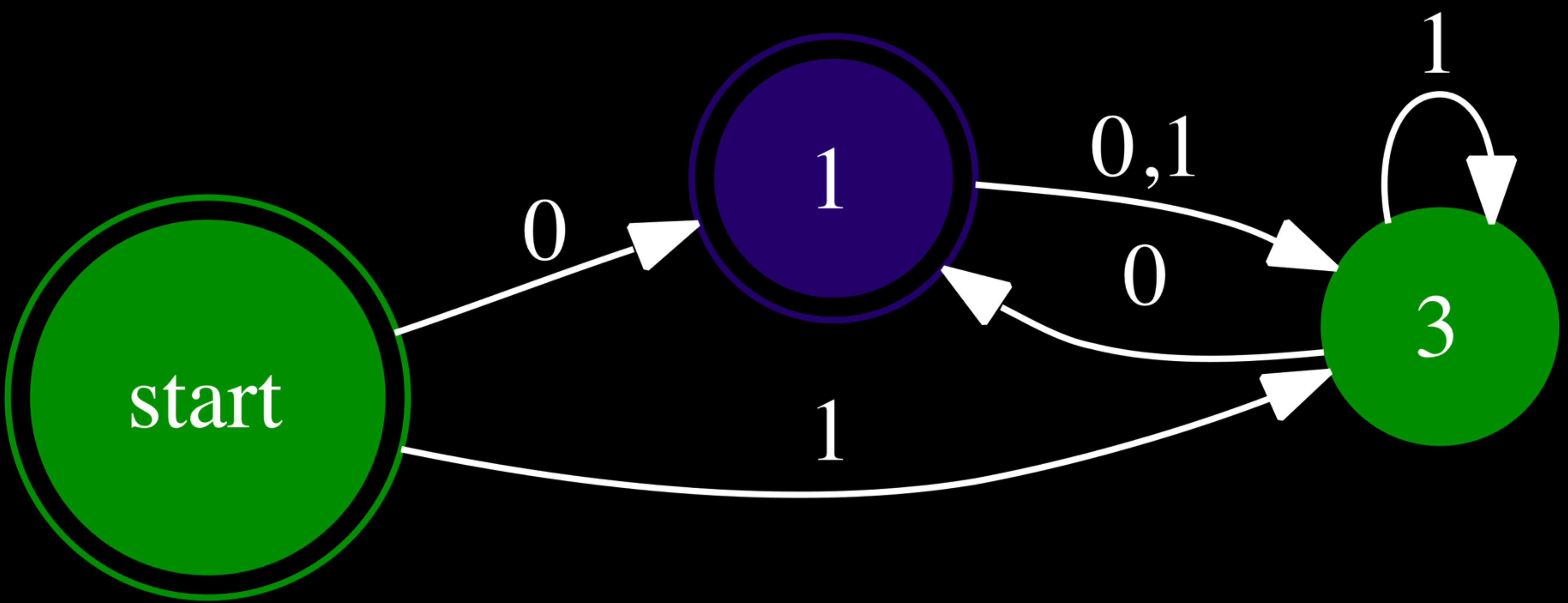


Showing Myhill-Nerode equivalence classes

Small Example – classically minimized

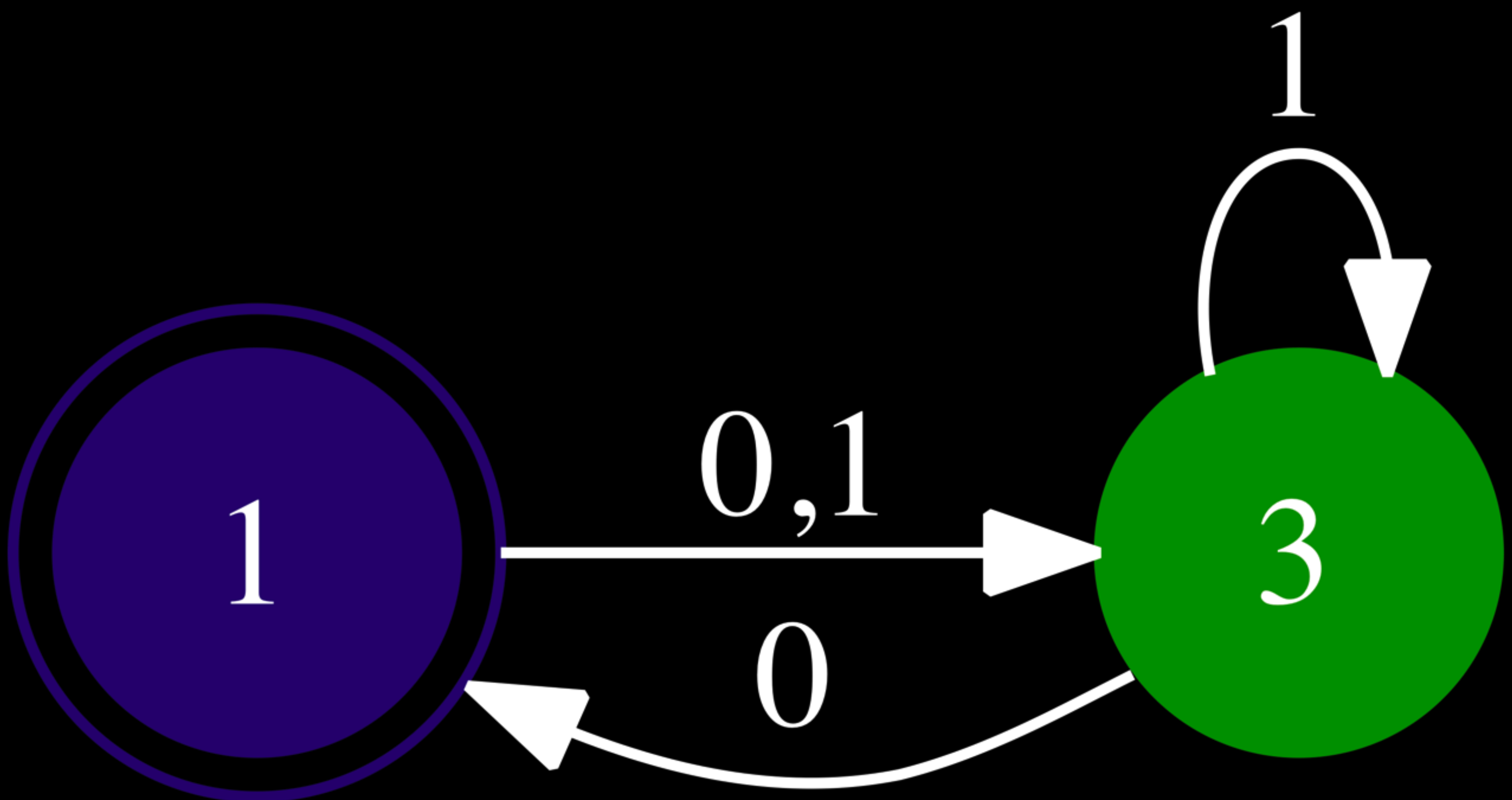


Small Example – classically minimized



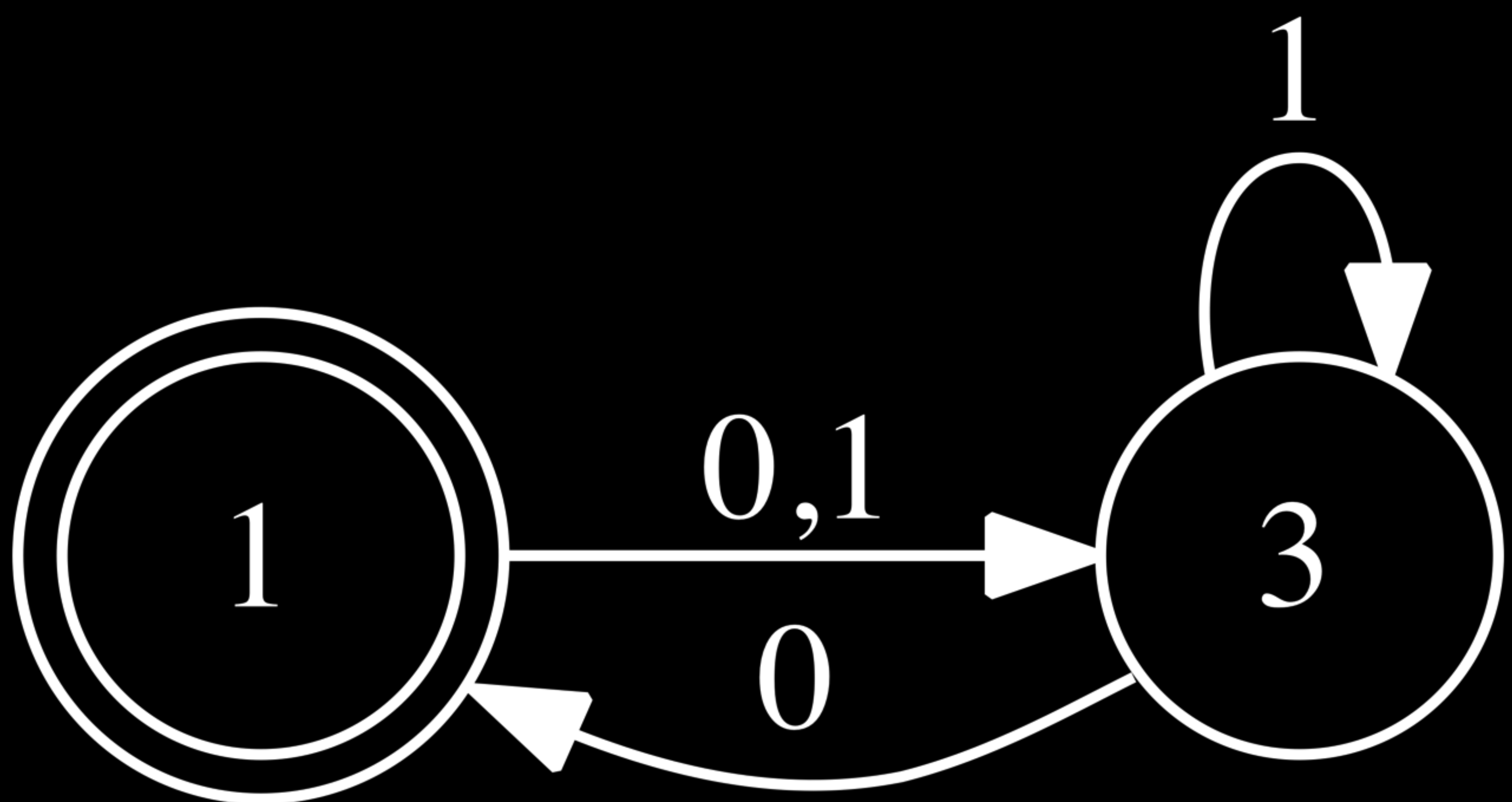
Showing f-equivalence classes

Small Example – hyper-minimized



Showing f-equivalence classes

Small Example – hyper-minimized

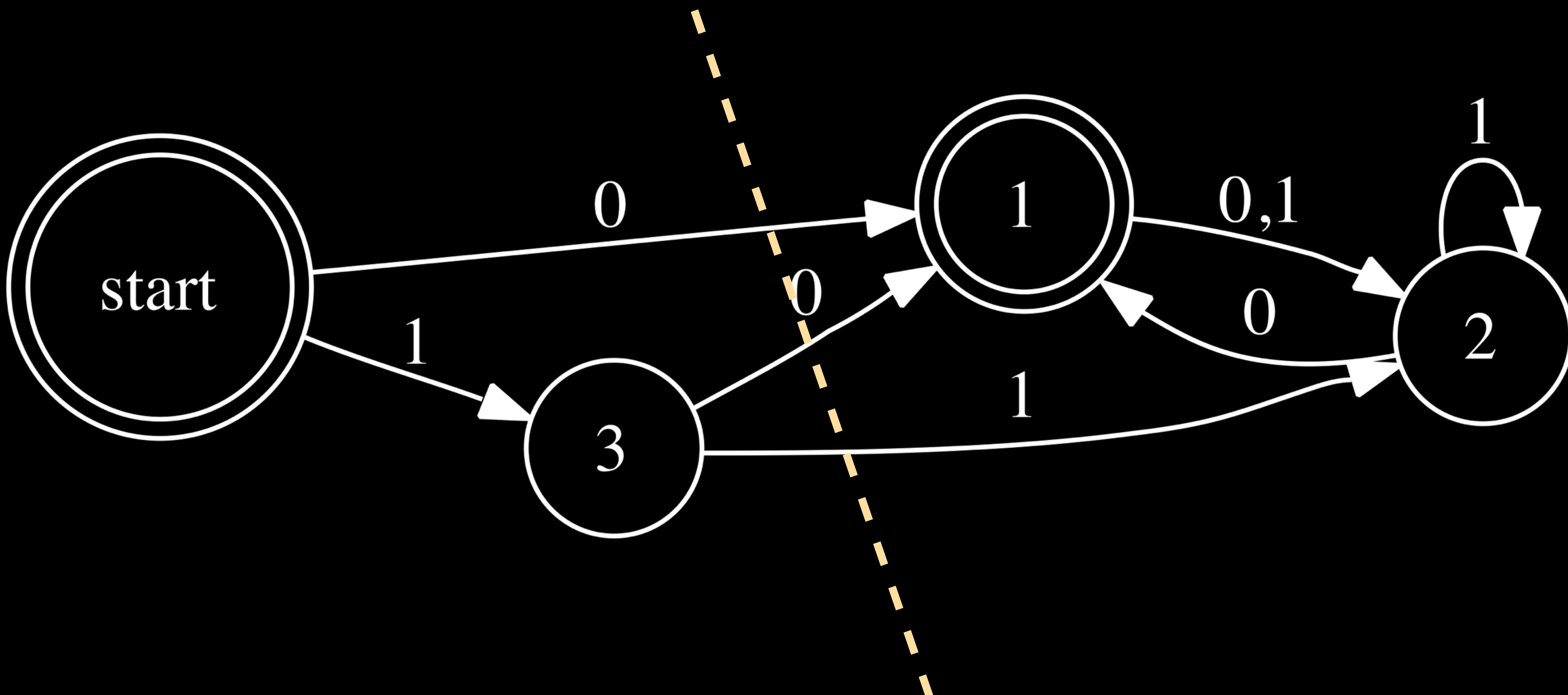


Elementary properties

- Let q_1 be a state from DFA D_1, and q_2 be a state from D_2 . If $q_1 \sim q_2$, then for any input c, $\delta(q_1 , c) \sim \delta(q_2 , c)$.

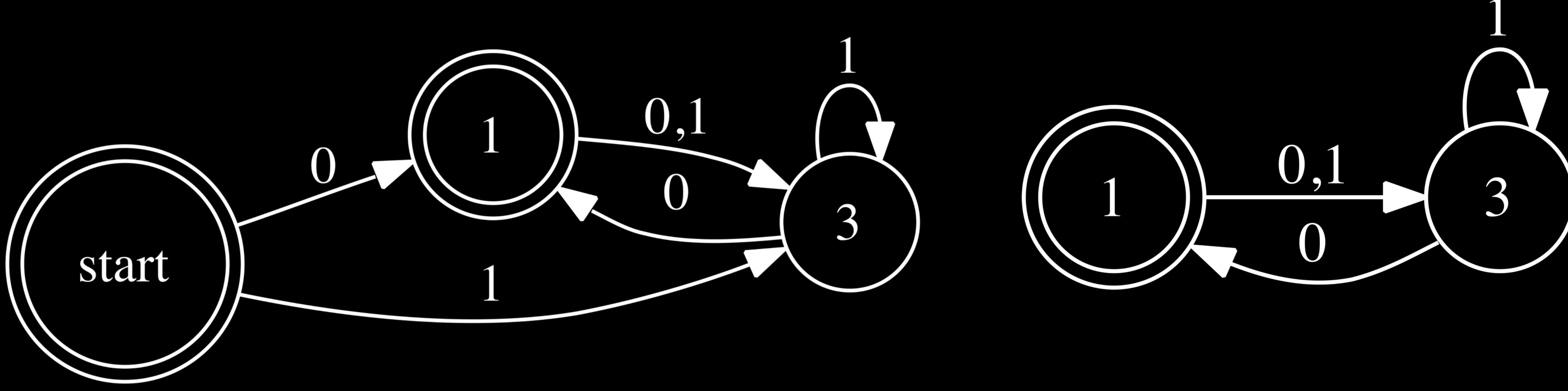
- If $D_1 \sim D_2$, then $\forall q_1 \in Q_1 , \exists q_2 \in Q_2 : q_1 \sim q_2$.

Preamble and Kernel



Kernel isomorphism

If $D_1 \sim D_2$ and both are classically minimized, then their kernels are isomorphic.



Preamble isomorphism

If $D_1 \sim D_2$ and both are hyper-minimized, then their preambles are (somewhat) isomorphic. These aspects within the preamble may differ:

- Whether a preamble state is accepting or not.
- Transitions from the preamble to the kernel can move within an f-equivalence class.

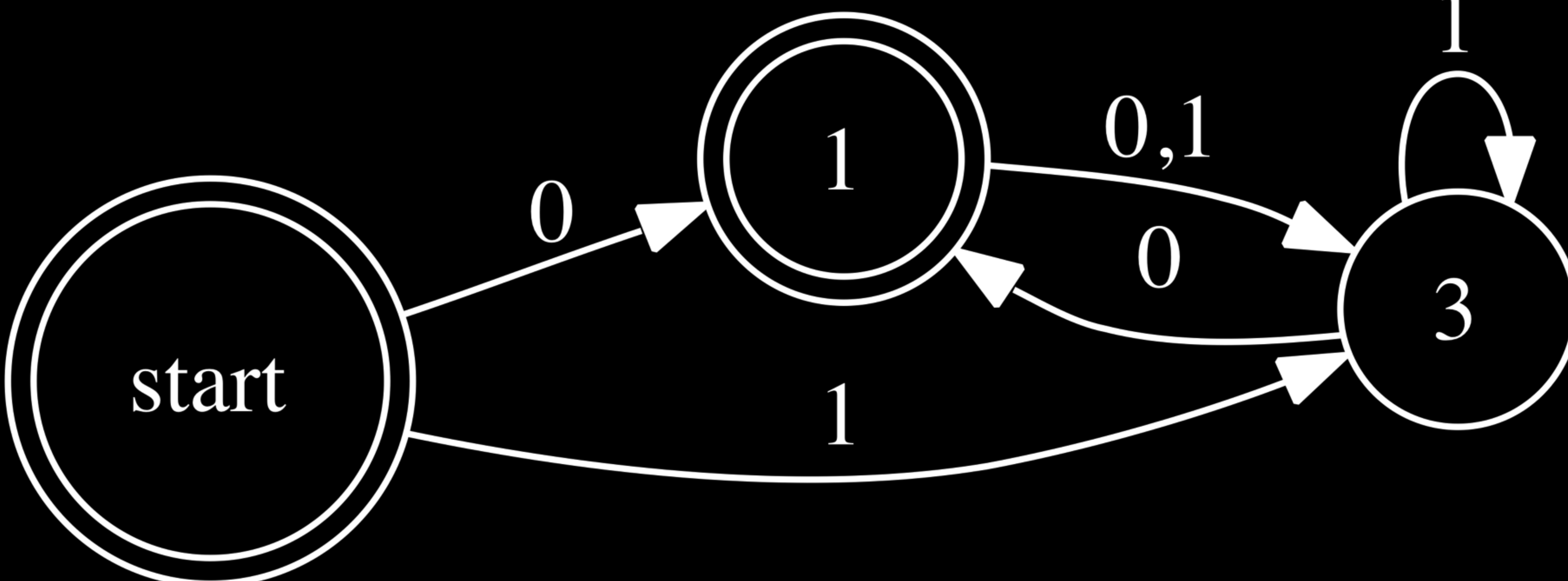
Minimization Algorithm

Classical Minimization
1. Delete unreachable states 2. Find equivalent states 3. Merge states within each equivalence class

Minimization Algorithm

Classical Minimization	Hyper-Minimization
1. Delete unreachable states	1. (Classically) Minimize
2. Find equivalent states	2. Find equivalent states
3. Merge states within each equivalence class	3. Merge states within each equivalence class

1. Classically minimal



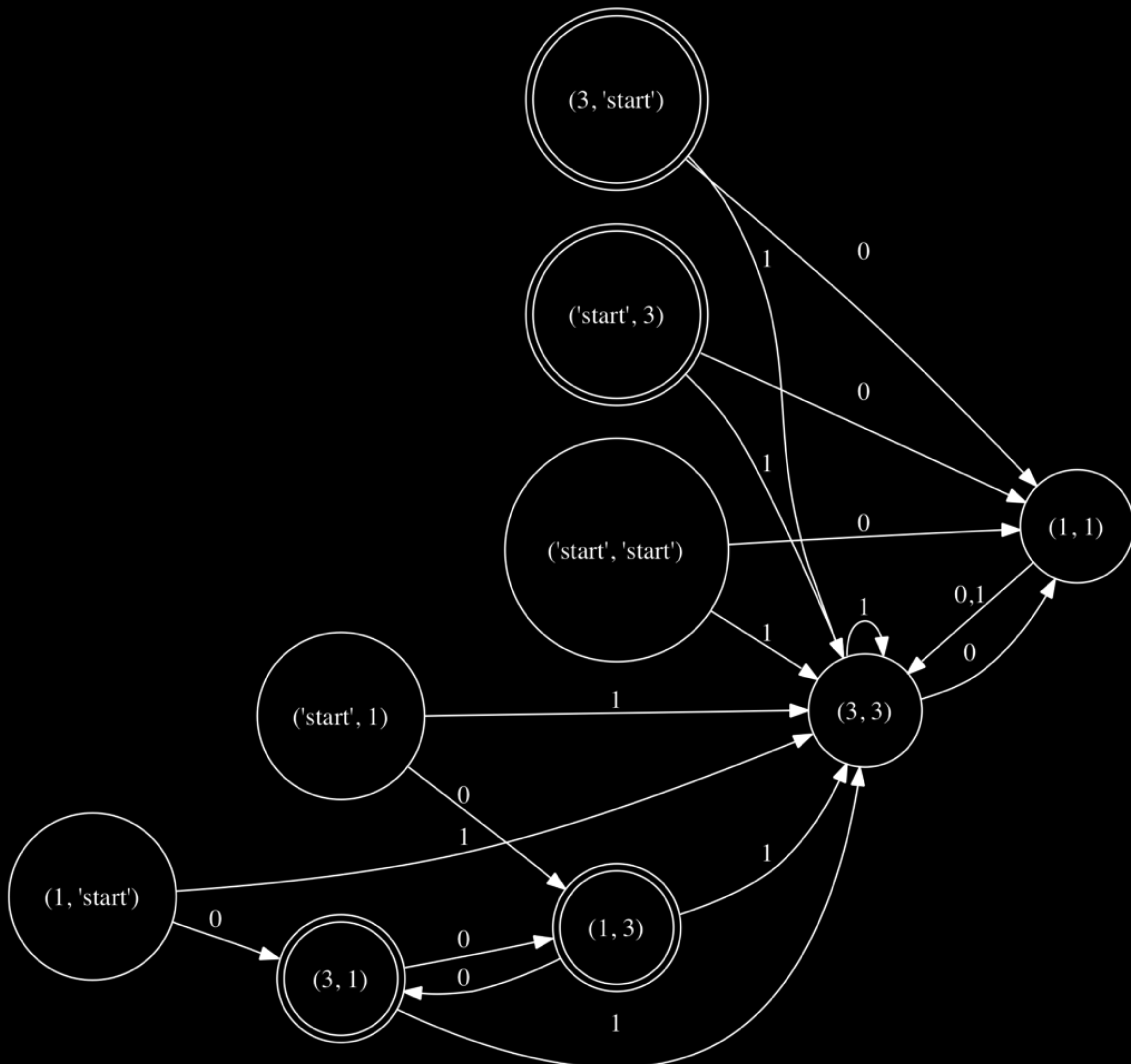
2. Finding f-equivalent state classes

i. Let $D_\otimes = D \otimes D$ be the standard DFA cross-product construction for symmetric difference.

ii. Find all states (q_0, q_1) in $D_\otimes$ which induce a finite language – q_0 and q_1 are f-equivalent in D.

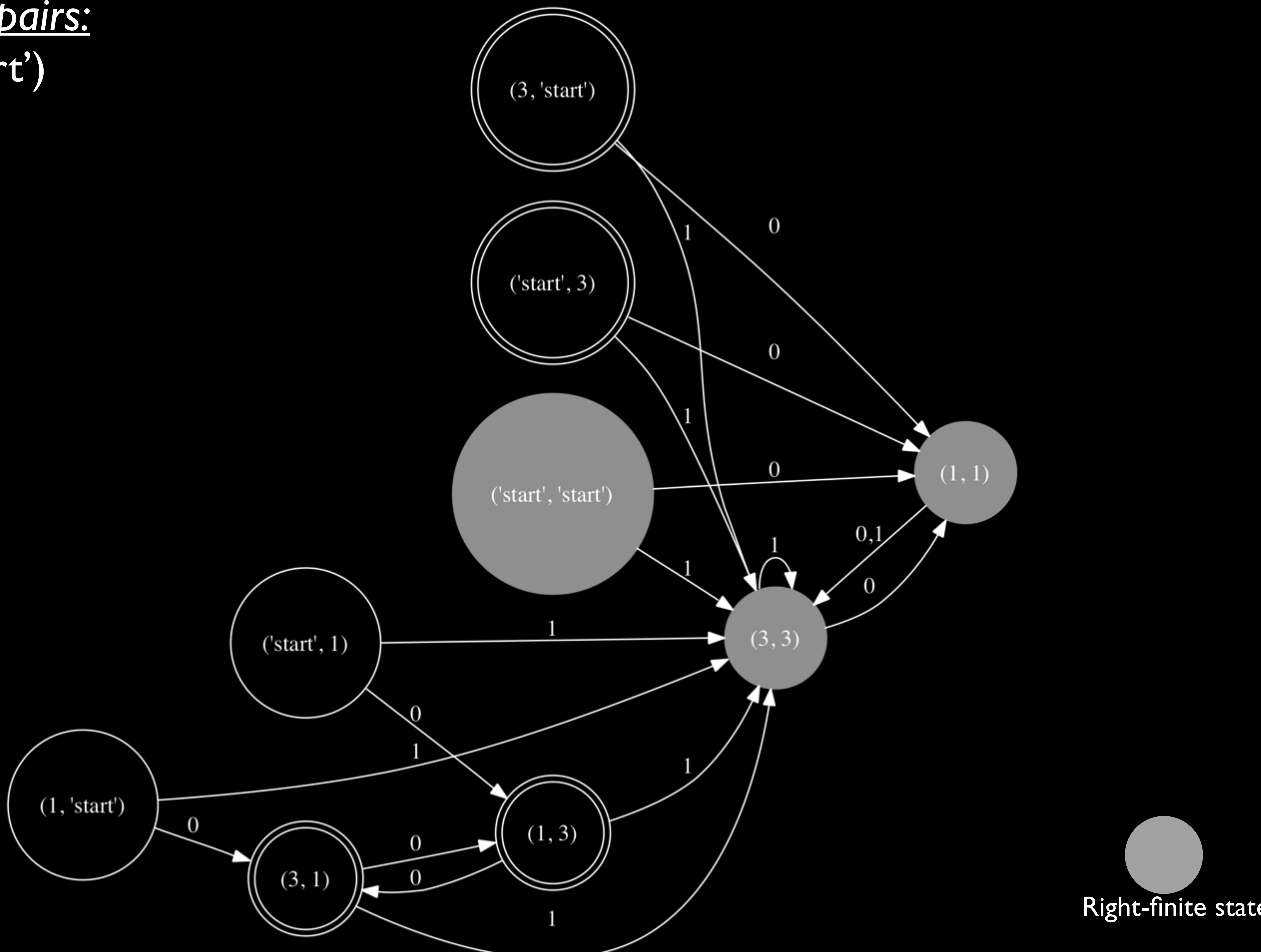
iii. Use the list of these pairs to construct the equivalence classes.

2.i. Cross-product with self



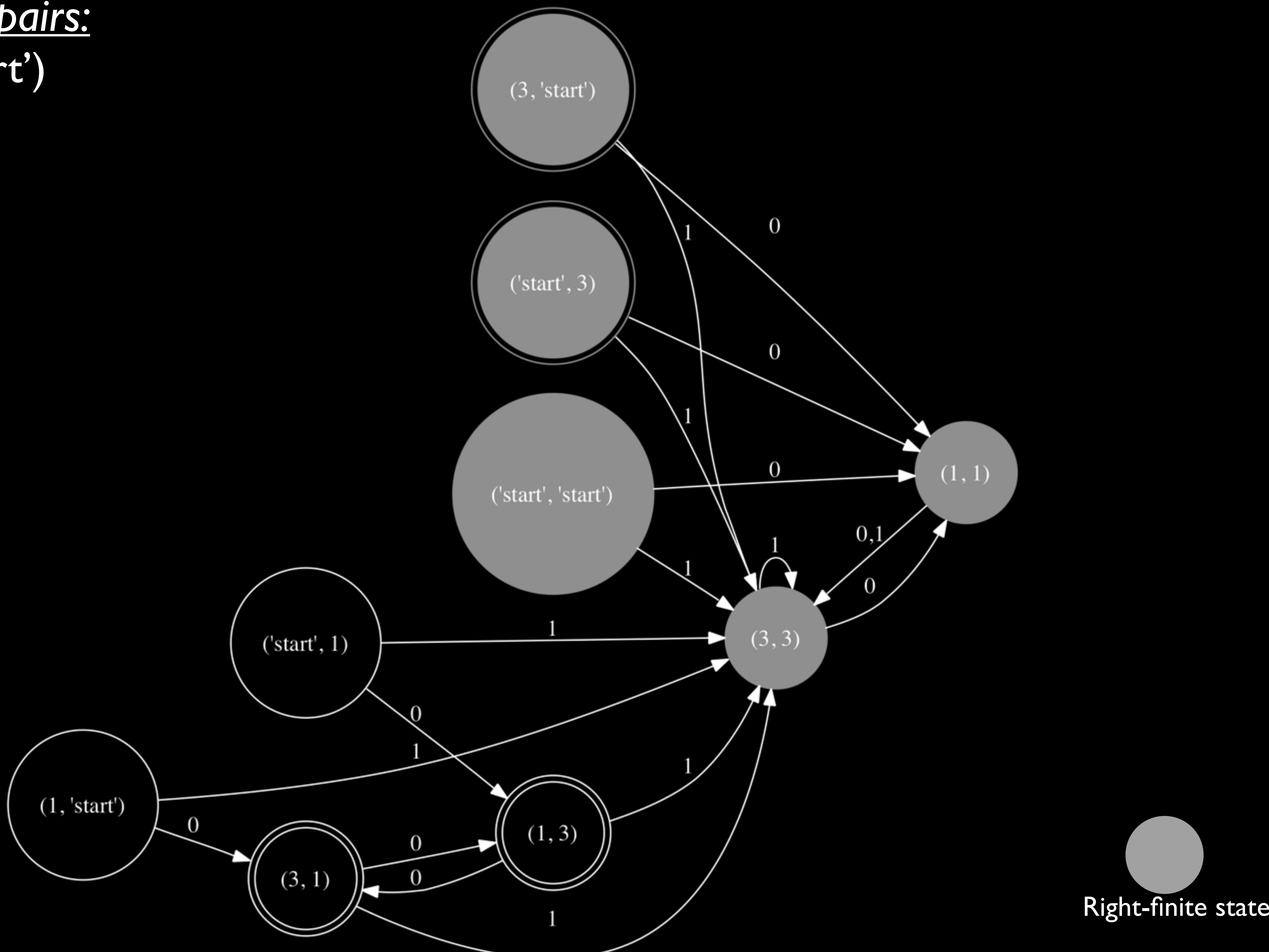
2.ii. Find all right-finite states

Equivalent pairs:
(‘start’, ‘start’)
(1, 1)
(3,3)



2.ii. Find all right-finite states

<u>*Equivalent pairs:*</u>
('start', 'start')
(1, 1)
(3,3)
(3, 'start')
('start', 3)



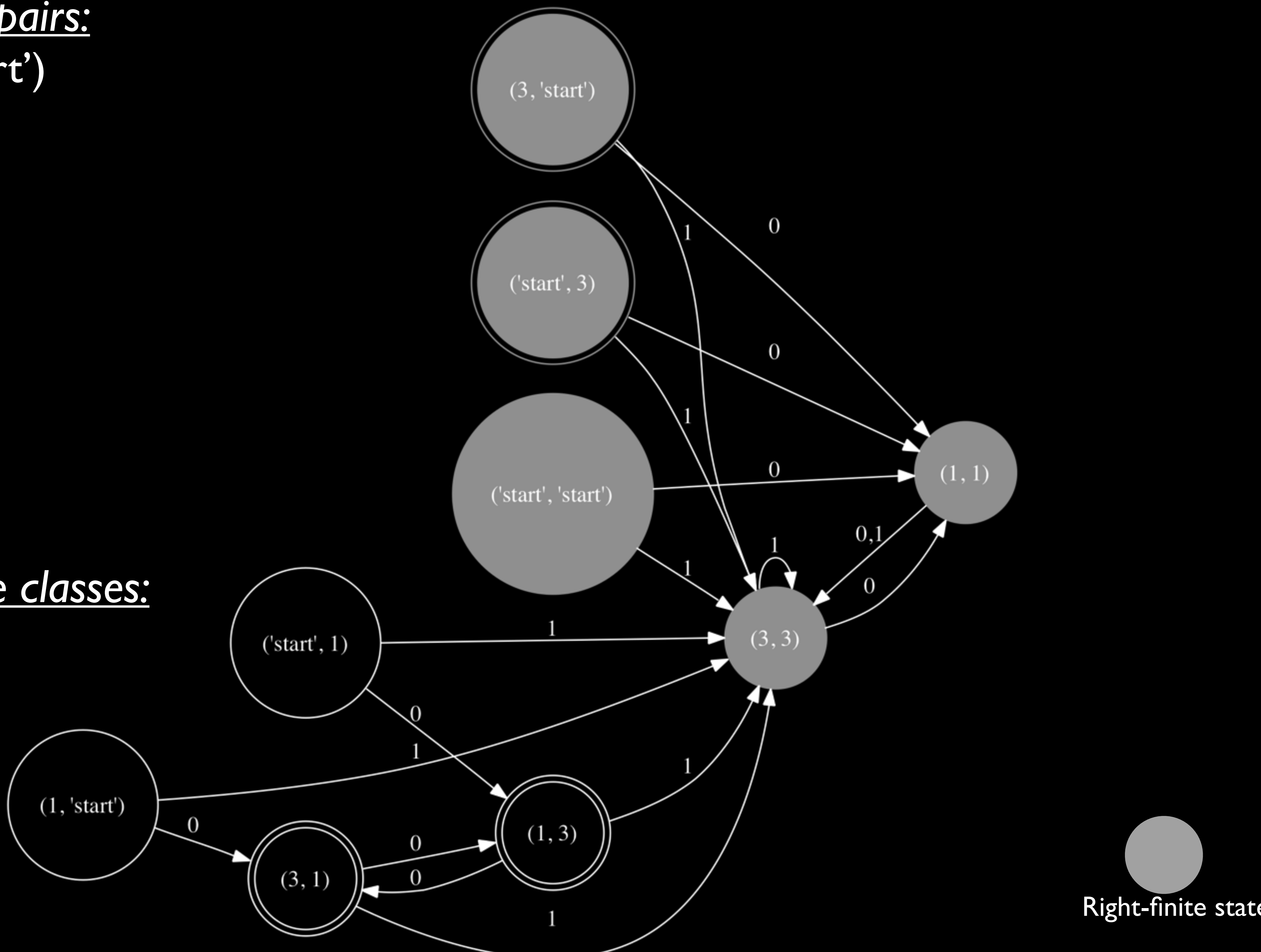
2.iii. Construct equivalence classes from pairs

Equivalent pairs:

(‘start’, ‘start’)
(1, 1)
(3,3)
(3, ‘start’)
(‘start’, 3)

Equivalence classes:

{‘start’, 3}
{1}

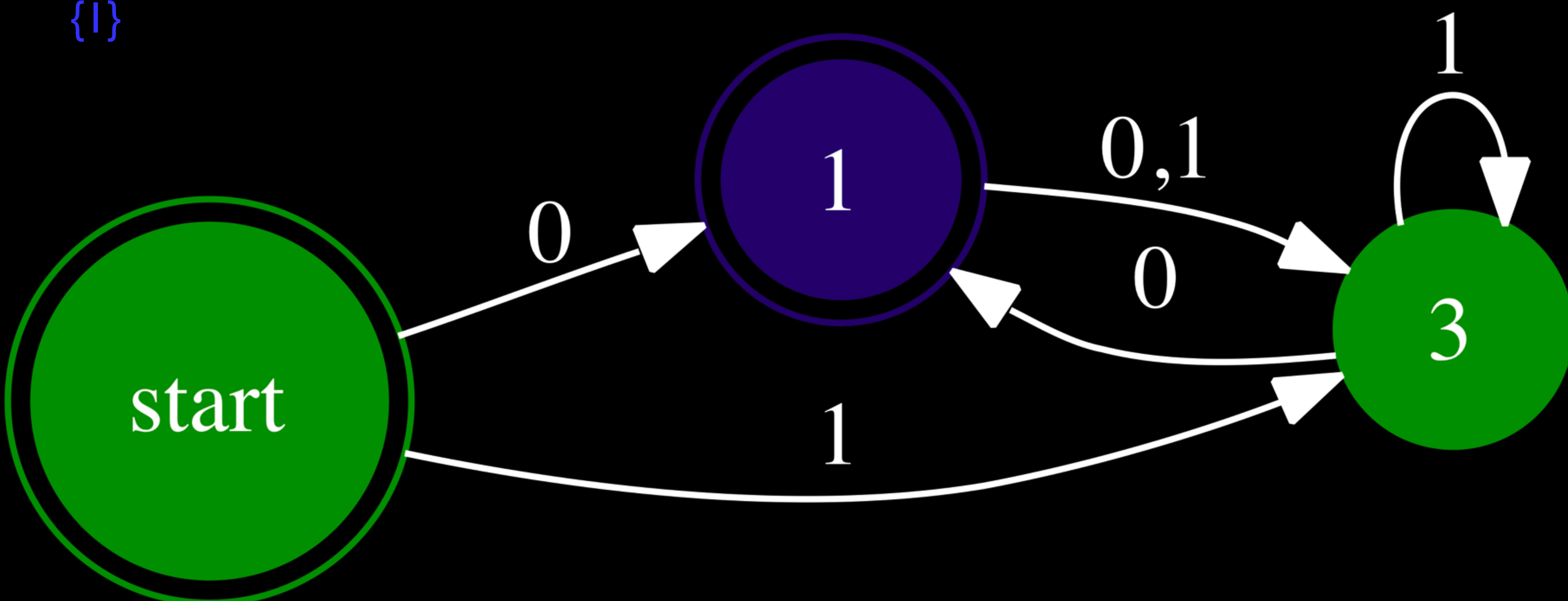


2.iii. Construct equivalence classes from pairs

Equivalence classes:

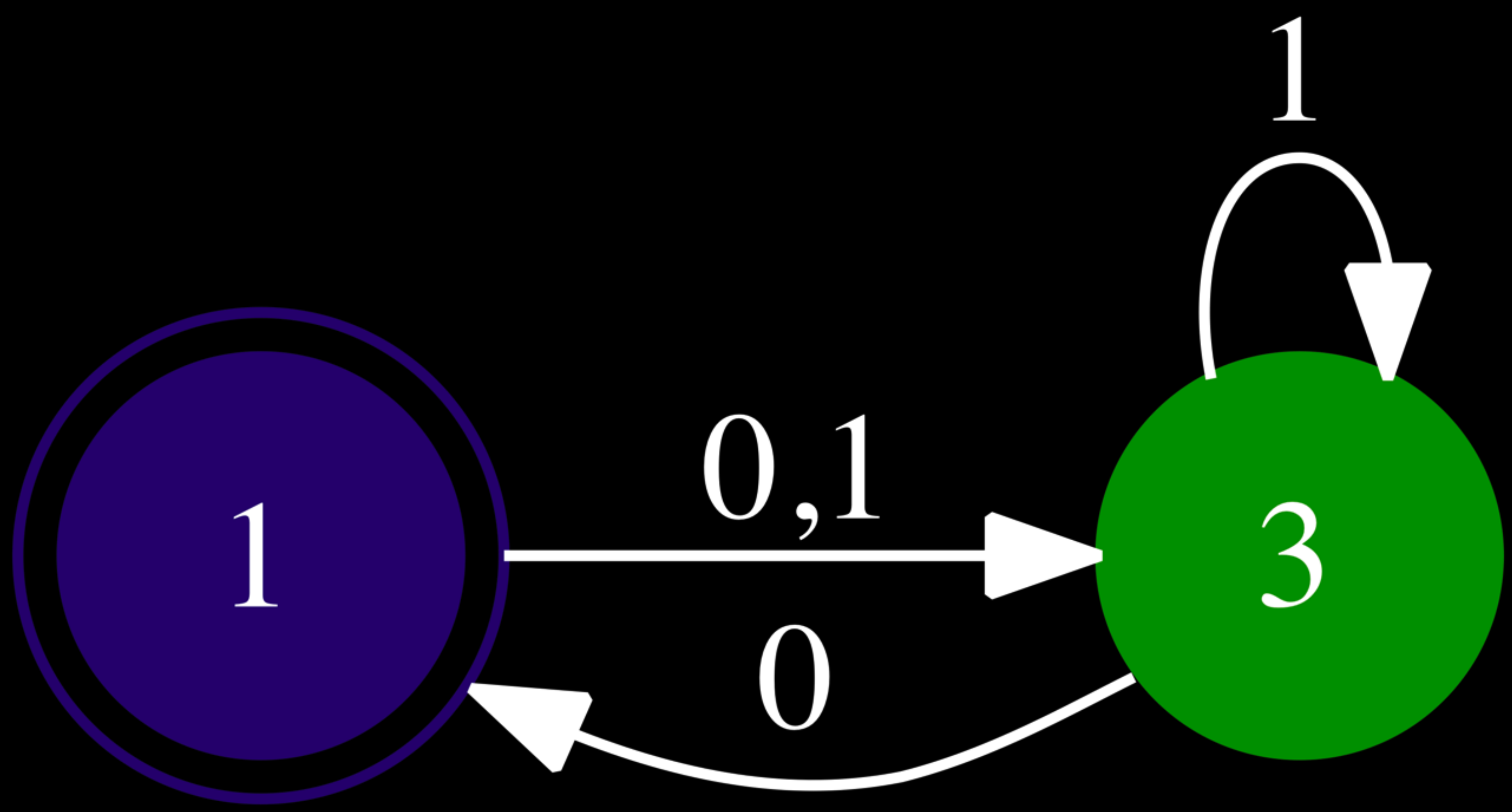
{'start', 3}

{1}



Showing f-equivalence classes

3. Merge states within each equivalence class



Showing f-equivalence classes

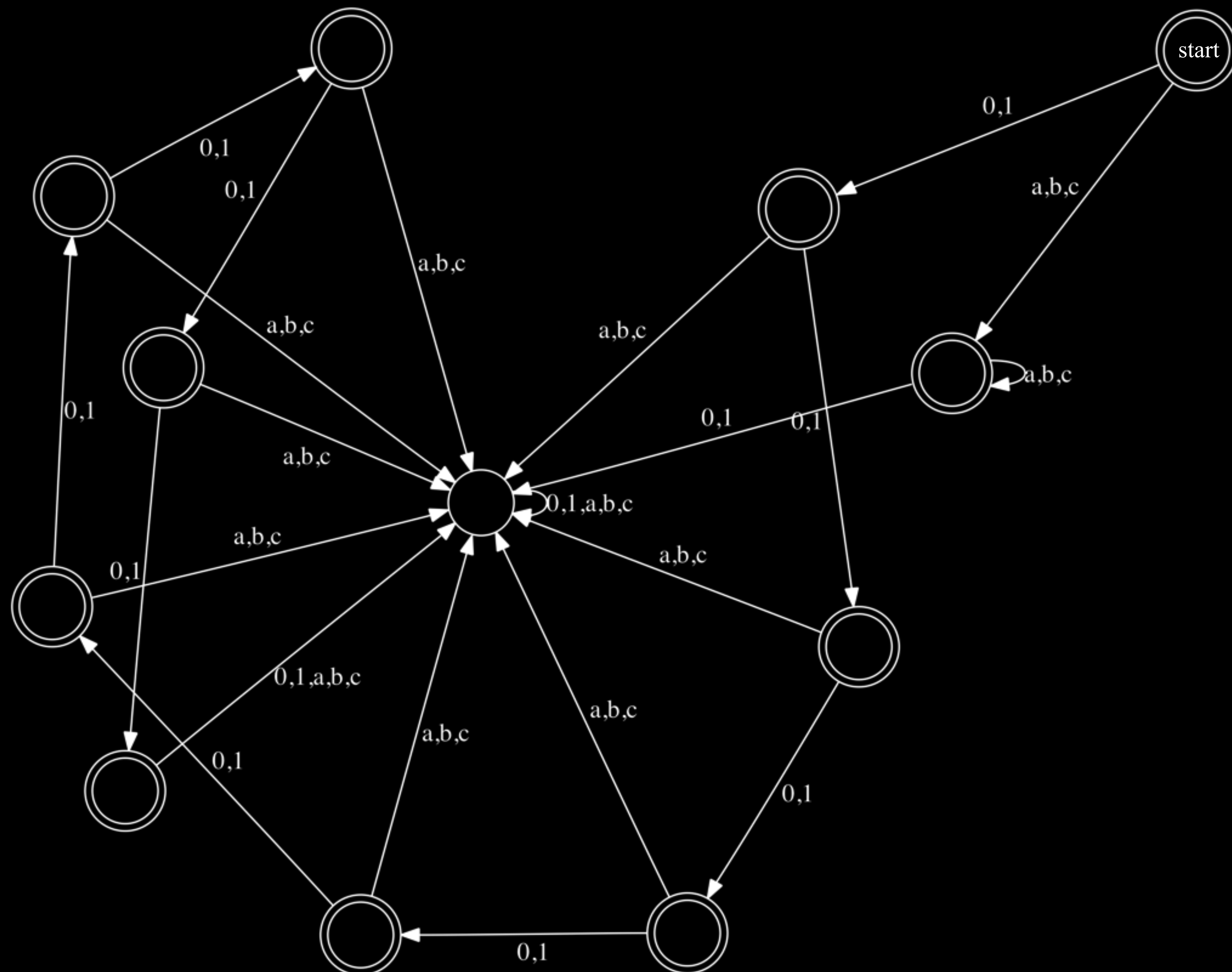
Finite-Factoring

- Motivation: hyper-minimization changes the language
- Use hyper-minimization to split a regular language into two parts: infinite and finite
- Use a DFCA to recognize the finite part
- What is a DFCA?

Finite-Factoring Algorithm

1. Let D' = *hyper_minimize(D)*
2. Let D_f = *xor_cross_product(D, D′)*
3. Let $n = max(|w| : w \in L(D_f))$
4. Minimize the DFCA (D_f, n)
5. Return $(D', (D_f, n))$

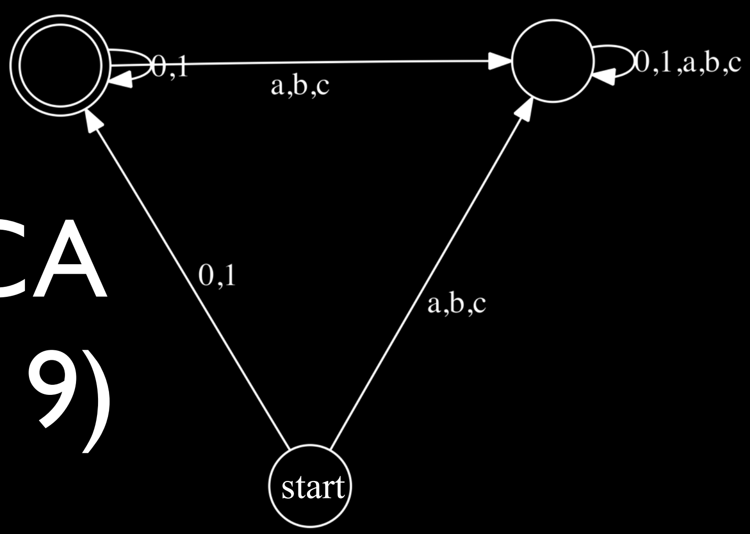
Finite-Factoring



Finite-Factoring

DFA

DFCA
(n = 9)



Open Problems, Questions

Source code: http://ianab.com/hyper/

Thanks: Lenny Pitt, Ian Shipman, Viliam Geffert,
Python, Keynote, Graphviz