Enforcing Detectability in Discrete Event Systems via Adaptive Control Sequences

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Magister Scientice in Computer Engineering

Martha Christou

The University of Cyprus, Nicosia

Department of Electrical and Computer Engineering

April 2021

Επιβολή Ανιχνευσιμότητας σε Συστήματα Διακριτών Συμβάντων Μέσω Προσαρμοστικών Ακολουθιών Ελέγχου

Διατριβή που κατατίθεται ως μέρος των απαιτήσεων για το πτυχίο
Μάστερ στην Μηχανική Υπολογιστών

Μάρθα Χρίστου

Πανεπιστήμιο Κύπρου, Λευκωσία
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Apr'ilios 2021

THESIS APPROVAL SHEET

Thesis Title: *Enforcing Detectability in Discrete Event Systems via Adaptive Control Sequences*

Student's Name: Martha Christou

We the members of the Advisory Committee for the above named student verify that the thesis satisfies the requirements of the Graduate School as approved by the Graduate Faculty.

Christoforos N. Hadjicostis
Professor
Department of Electrical and Computer Engineering
Director of Thesis

Charalambos D. Charalambous
Professor
Department of Electrical and Computer Engineering
Dean, School of Engineering

Dr. loannis Tzortzis
Post Doctoral Researcher
Department of Electrical and Computer Engineering

Advisory Committee

Date

Acknowledgement

I would like to express my gratitude to my advisor Professor Christoforos Hadjicostis for his patience and his continuous encouragement and support. I appreciate his extensive knowledge and expertise in many areas and his nice treatment and assistance throughout my studies at the University of Cyprus.

Big thanks also go to my family and my friends for the support they provided throughout my entire life. Without their love, patience and encouragement I would not have finished this work.

Abstract

This thesis studies a class of problems in which we are given the model of a system, with an unknown (or partially known) initial state, and the goal is to apply a carefully chosen sequence of inputs so that, along with the observations (outputs) that are generated, one can determine exactly the current or initial state of the system. The main solution for this type of problems involves current- or initial-state estimation based on recorded sequences of observable events or outputs. Having recorded a sequence of observable events, typical estimation/inference tasks involve the determination of the exact current/initial state of the system or, more generally, the deduction of useful information about the possible current/initial states of the system, and/or the occurrence of certain (unobservable) events of interest. This can be key for fault diagnosis and other event inference tasks in discrete event systems, or even for supervisory control strategies that aim at achieving various objectives (e.g., opacity enforcement or deadlock avoidance).

Πρόλογος

Το detectability enforcement είναι μια διαδικασία που γίνεται σε συστήματα διακριτών συμβάντων (discrete event systems (DES)),τα οποία μπορούν να μοντελοποιηθούν ως finite automata with outputs δηλαδή με πεπερασμένο αριθμό από καταστάσεις (states) και εξόδους.

Σε αυτή τη διατριβή, μελετούμε το detectability (δηλαδή, την ενδεχόμενη ακριβή γνώση της τρέχουσας κατάστασης του συστήματος) και την επιβολή του σε non-deterministic finite automata με ελεγχόμενες εισόδους (πλήρως καθορισμένες σε κάθε καταστάση (fully defined) αλλά χωρίς πλήρη γνώση της αρχικής κατάστασης (initial state).

Στην εργασία αυτή ενδιαφερόμαστε για εποπτικές στρατηγικές ελέγχου (supervisory control), όπου η ακολουθία εισόδων επιλέγεται προσεκτικά έτσι ώστε η ακολουθία εξόδων που δημιουργείται από το σύστημα (μαζί με την εφαρμογή της γνωστής ακολουθίας εισόδων), να μπορεί να χρησιμοποιηθεί για τον προσδιορισμό της ακριβούς κατάστασης του συστήματος στο τέλος της διαδικασίας (state detectability, κατάσταση ανιχνευσιμότητας).

Οι στρατηγικές που θα χρησιμοποιηθούν είναι προσαρμοστικές υπό την έννοια ότι ο controller επιτρέπεται να βασίζει την απόφαση του ως προς το ποια είσοδος θα εφαρμοστεί στη συνέχεια σε μια συγκεκριμένη στιγμή, με βάση την ακολουθία των εισόδων που έχουν ήδη εφαρμοστεί, και την ακολουθία των εξόδων που έχουν παρατηρηθεί μέχρι αυτό το σημείο.

Αυτό οδηγεί σε μια διαμόρφωση παιχνιδιού δύο παικτών, όπου ο controller του συστήματος επιλέγει την είσοδο για εφαρμογή, ενώ το σύστημα 'επιλέγει' μια έξοδο (πιο συγκεκριμένα, το σύστημα παράγει μια έξοδο μέσα από το σύνολο των εφικτών εξόδων). Στη διατριβή αυτή αναλύεται πως η στρατηγική ελέγχου μπορεί να επιτευχθεί συστηματικά χρησιμοποιώντας μια αλγοριθμική περιγραφή της προκύπτουσας δομής παιχνιδιού. Γίνεται επίσης συζήτηση για τις βέλτιστες (με ελάχιστο κόστος) στρατηγικές ελέγχου που ελαχιστοποιούν το χειρότερο (μέγιστο) μήκος της ακολουθίας των εισόδων ελέγχου που πρέπει να εφαρμοστούν για να

φτάσουμε σε κατάσταση ανιχνευσιμότητας (state detectability).

Contents

Acknowledgement iii

Abstract iv

Πρόλογος v

1 Introduction and Motivation 1

2 Basic Definitions, Background and Notation 4

2.1 Deterministic Finite Automata Under Partial Observation 4

2.2 Fully Defined Nondeterministic Finite Automata with Outputs 7

2.3 Strong Detectability and its Verification 10

2.4 K-Detectability and its Verification 11

3 Problem Formulation and Solution Approach 14

4 Min-Max Strategies 24

5 Conclusions 33

5.1 Summary 33

5.2 Future Work 34

List of Figures

2.1 Nondeterministic finite automaton (NFA) with outputs. 7
2.2 Construction of the observer for the NFA in Fig. 2.1 based on inputs only. 8

3.1 Problem formulation 15
3.2 Observer construction based on inputs and outputs. 16
3.3 Illustration of Rule 1. 18
3.4 llustration of Rule 2. 18
3.5 Solution approach. 19
3.6 Nondeterministic finite automaton (NFA) with outputs. 20
3.7 Construction of the observer with inputs and outputs for the NFA in Fig. 3.6. . . . 21
3.8 Solution approach for NFA with outputs in Fig. 3.6 22

4.1 Acyclic Digraph. 25
4.2 FDNFA considered in Example 2. 26
4.3 FDNFA considered in Example 2. 28
4.4 Observer construction for the system in Fig. 4.3 29
4.5 Feasible Strategies for the system in Fig. 4.3. 30
4.6 Cost observer of NDNFA Fig. 3.6. 31
4.7 Possible strategies for the system in Fig. 3.7. 32

Chapter 1

Introduction and Motivation

The property of interest in this thesis is the notion of *detectability* [8, 9]: a given system is called *detectable* if, after any sequence of observations of sufficient length, the current state of the system can be isolated (i.e., it becomes perfectly known). More specifically, a system is *strongly detectable* if, after observing any sufficiently long sequence of observations that can be generated by the system, the current state of the system can be pinpointed exactly. In the case of discrete event systems (DES) that can be modeled as (deterministic or nondeterministic) finite automata under partial observation, the property of detectability can be verified by constructing the *observer* of the system (with complexity exponential in the number of states of the given finite automaton) or by constructing the *detector* of the system (with complexity polynomial in the number of states of the given finite automaton) [1, 8, 9]. Detectability has been generalized to K-detectability to capture the ability to isolate the state of the system within a set of cardinality at most K, following the occurrence of any sequence of observations of sufficient length [10]. Extensions that consider how to enforce detectability via supervisory control strategies have also been considered (see, for example, [11]).

The above works consider primarily DES that can be modeled as nondeterministic finite automata with events that are partially observable (e.g., some of the events may be unobservable). In this thesis, we consider DES that can be modeled as *nondeterministic* finite automata with inputs (fully defined at each state) and outputs. In other words, each time an input is applied to the system, one also observes an output that is generated by the system (and which can be used to refine any estimate of the current/initial state of the system). This implies that, apart from the sequence of inputs that is applied to the system, the sequence of outputs that is produced is also important for

state identification.

Sequences of inputs that can be used for state estimation/classification have been studied mostly in the context of *finite state machines* (i.e., deterministic finite automata with outputs) and come under various names, depending on what they achieve.

- A *synchronizing* sequence is a sequence of inputs that guarantees to lead the system to a known state. In other words, despite any uncertainty in the initial state of the system, the system will be driven to a specific final state after the application of the synchronizing sequence [12] , [13].
- A *homing* sequence is a sequence of inputs that allows us to determine, based on the sequence of outputs that is generated, the final state of the system [2]. The main difference of a homing sequence from a synchronizing sequence is that it also takes into account the sequence of outputs that is generated (i.e., a homing sequence requires access to the outputs of the system, whereas a synchronizing sequence does not require such access). Note that a homing sequence is called *adaptive* if one is allowed to choose the next input based on the sequence of outputs generated so far; if this is not the case (i.e., if the sequence of inputs is chosen *a priori*, without taking into account any of the outputs that are observed), the homing sequence is called *preset*.
- A *distinguishing* sequence is a sequence of inputs that allows one to uniquely identify the initial state by observing the output sequence [14]. In the case of deterministic finite automata (which was the focus of most of the earlier work in this area), a preset (or adaptive) distinguishing sequence is necessarily a preset (or adaptive) homing sequence.

Most works that focus on the type of problems described above (i.e., choosing appropriate input sequences for current/initial state identification) assume that the underlying system is a deterministic finite automaton with inputs and outputs, also referred to as a finite state machine (FSM). Another common assumption for this type of problems is that the given machine is minimal, connected and fully (or completely) specified (i.e., each input is defined at each state). Challenges include the fact that the initial state is unknown (or may be known to belong to a subset of states) and the output that is observed can only partially identify activity in the system. Discussions on existing work on the above topics can be found in [1, 15].

In this thesis, we consider the problem of obtaining an adaptive homing sequence for an underlying

nondeterministic finite automaton with outputs. More specifically, we are interested in strategies where the sequence of inputs is carefully chosen so that the sequence of outputs that is generated by the system can be used to determine the exact state of the system at the end of the process (state detectability). The strategies are adaptive in the sense that the controller is allowed to decide what input to apply at a particular instant of time *based* on the sequence of inputs that has already been applied and the sequence of outputs that has been observed up to that point. Compared to traditional approaches on synchronizing/homing/distinguishing sequences (see, for example, [15] and the more recent work in [12, 13, 16, 17]), our work generalizes the underlying setting by considering nondeterministic finite automata with outputs (and inputs that are fully defined at each state). Compared to work on detectability (see, for example, [8, 9] but also the work in [11] that aims at enforcing detectability), our work generalizes the existing framework by considering nondeterministic finite automata with outputs. The presence of outputs naturally leads to minimax strategies in two-player game formulations, in which the controller chooses the input to apply whereas the system "chooses" an output (more precisely, the system produces an output among the set of feasible outputs).

The authors of [11] also deal with this class of problems but they assume (unlike our work in this thesis) that they can control the input/output pair (i.e., they can instruct the system to execute, among transitions associated with a specific input, the ones associated with a chosen output). In our case, however, we can only control the event (input), but not the output. Our goal is also slightly different because we aim to drive the system to a situation where we know its state exactly. However, we could also adopt an objective that is closer to the work in [11], which aims at operating the system in a region where the state of the system is known exactly. More specifically, we could aim for an adaptive strategy (i.e., a strategy that reacts on the output seen) so that the system is kept in detectable states all the time (regardless of what outputs the system chooses), at least from a certain point onwards. We elaborate on this issue later on, once we have the opportunity to describe the setting and the proposed approach in more detail.

Another paper that deals with related problems is the work in [16], which considers the case when there is no output. Thus, the authors of [16] aim at finding the minimal sequence of inputs, under which the system reaches a known state. In contrast, the uncertainty introduced by the presence of outputs in our setting implies that we need to employ a min-max approach: since there is uncertainty as to what output the system will generate, we choose to obtain the minimal length sequence of inputs, under the worst possible scenario of outputs provided by the system.

Chapter 2

Basic Definitions, Background and Notation

2.1 Deterministic Finite Automata Under Partial Observation

Let Σ be an alphabet (set of events) and denote by Σ^* the set of all finite-length strings of elements of Σ (sequences of events), including the empty string ε (the length of a string s is denoted by $|s|$ with $|\varepsilon| = 0$). A language $L \subseteq \Sigma^*$ is a subset of finite-length strings in Σ^* [7] (i.e., sequences of events with the convention that the first event appears on the left). Given strings $s,t \in \Sigma^*$, the string st stands for the concatenation of s and t, which denotes the sequence of events captured by s followed by the sequence of events captured by t. For a string s, $\bar{s}$ denotes the *prefix-closure* of s, and is defined as $\bar{s} = \{t \in \Sigma^* \mid \exists t' \in \Sigma^* \{tt' = s\}\}$.

Definition 2.1.1. *(Deterministic Finite Automaton (DFA)) A deterministic finite automaton (DFA) is captured by $D = (Q, \Sigma, \delta, Q_0)$, where $Q = \{q^{(1)}, q^{(2)}, \ldots, q^{(|Q|)}\}$ is the finite set of states, Σ is the finite set of inputs, and Q_0, $Q_0 \subseteq Q$, is the set of possible initial states. The possibly partially defined transition function $\delta : Q \times \Sigma \to Q$ specifies, for a state $q \in Q$ and an input $\sigma \in \Sigma$, the next state $q' \in Q$ that the system transitions to. This is denoted by $\delta(q,\sigma) = q'$; $\delta(q,\sigma)$ is undefined if input σ is not feasible at state q.*

The function δ can be extended from the domain $Q \times \Sigma$ to the domain $Q \times \Sigma^*$ in the routine recursive manner:

$$\delta(q, \sigma s) = \begin{cases} \delta(\delta(q,\sigma), s), & \text{if } \delta(q,\sigma) \text{ is defined,} \\ \text{undefined,} & \text{otherwise,} \end{cases}$$

for $q \in Q$, $s \in \Sigma^*$, and $\sigma \in \Sigma$ (note that we take $\delta(q,\varepsilon) := q$ for all $q \in Q$). With this notation at hand, the behavior of *DFA D* is captured by $L(D) := \{s \in \Sigma^* \mid \exists q_0 \in Q_0 \{\delta(q_0, s) \text{ is defined}\}\}$.

Definition 2.1.2. *(Nondeterministic Finite Automaton (NFA)) A nondeterministic finite automaton (NFA) is captured by* $N = (Q, \Sigma, \delta, Q_0)$*, where* $Q = \{q^{(1)}, q^{(2)}, \ldots, q^{(|Q|)}\}$ *is the finite set of states,* Σ *is the finite set of inputs, and* Q_0*,* $Q_0 \subseteq Q$*, is the set of possible initial states. The nondeterministic transition relation* $\delta \subseteq Q \times \Sigma \times Q$ *is such that* $(q, \sigma, q') \in \delta$ *if from state* q *with input* σ *we transition to state* q'*.*

Remark 2.1.1. *Note that the relation* δ *may include* $(q, \sigma, q') \in \delta$ *and* $(q, \sigma, q'') \in \delta$ *for different* q' *and* q''*. This implies that the transition from state* q *with input* σ *is nondeterministic in the sense that the system may end up in state* q' *or state* q'' *(or other states if there exist additional* $(q, \sigma, *) \in \delta$*). We can also sometimes define* δ_f *as a nondeterministic transition function:* $\delta_f(q, \sigma) = \{q' \in Q | (q, \sigma, q') \in \delta\}$*.*

Given an NFA $N = (Q, \Sigma, \delta, Q_0)$, we say that input σ, $\sigma \in \Sigma$, is *defined* at state q if there exists (at least one) $q' \in Q$ so that $(q, \sigma, q') \in \delta$. Moreover, we say that N is *fully (or completely) defined* on input set Σ if for each $\sigma \in \Sigma$ and each state $q \in Q$, input σ is defined at state q.

The transition relation δ can be extended from the domain $Q \times \Sigma \times Q$ to the domain $Q \times \Sigma^* \times Q$ as follows: For $s = \sigma_1 \sigma_2 ... \sigma_k$ we have $(q, s, q') \in \delta$ if $\exists\ q_1, q_2, .., q_{(k-1)}$ *such that* $(q, \sigma_1, q_1), (q_1, \sigma_2, q_2), ..., (q_{(k-1)}, \sigma_k, q') \in \delta$.

Regardless of whether we deal with a DFA or an NFA, a simple observation model is to assume that a subset of events Σ_o, $\Sigma_o \subseteq \Sigma$, can be observed; however, the remaining events $\Sigma_{uo} \equiv \Sigma \setminus \Sigma_o$ are assumed to be unobservable. In such case, the natural projection $P : \Sigma^* \to \Sigma_o^*$ can be used to map any trace executed in the system to the sequence of observations associated with it (generated by it). This projection is defined recursively as $P(\sigma s) = P(\sigma)P(s)$, $s \in \Sigma^*, \sigma \in \Sigma$, with

$$P(\sigma) = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma_o, \\ \varepsilon, & \text{if } \sigma \in \Sigma_{uo} \cup \{\varepsilon\}, \end{cases}$$

where ε represents the empty trace [7].

Definition 2.1.3. *(Possible states following a sequence of observations (*$R : 2^Q \times \Sigma_o^* \to 2^Q$*)) Suppose that NFA* $N = (Q, \Sigma, \delta, Q_0)$ *is known to be in a set of possible states* Q'*,* $Q' \subseteq Q$*; the set of all possible states after subsequently observing* $\omega \in \Sigma_o^*$ *is*

$$\begin{aligned} R(Q', \omega) \quad = \quad & \{q \in Q \mid \exists q' \in Q', \exists s \in \Sigma^*,\ s.t. \\ & P(s) = \omega \wedge (q', s, q) \in \delta\}\ . \end{aligned}$$

Definition 2.1.4. *(Observer or Current-State Estimator [7]) Given an NFA $N = (Q, \Sigma, \delta, Q_0)$ with set of observable events $\Sigma_o \subseteq \Sigma$ under the natural projection map P, the observer (or current-state estimator) is a deterministic finite automaton (DFA) $N_{obs} = (Q_{obs}, \Sigma_o, \delta_{obs}, Q_{0,obs})$, which can be constructed as follows:*

1. *Each state of N_{obs} is associated with a unique subset of states of the original NFA N (this means that $Q_{obs} \subseteq 2^Q$ has at most $2^{|Q|}$ states).*
2. *The initial state $Q_{0,obs}$ of N_{obs} is the unobservable reach of Q_0 (i.e., $Q_{0,obs} = UR(Q_0) = R(Q_0, \varepsilon)$).*
3. *From any state $q_{obs} \in Q_{obs}$ of the current-state estimator, the next state for any $\sigma_o \in \Sigma_o$ is captured by $\delta_{obs}(q_{obs}, \sigma_o) = R(q_{obs}, \sigma_o)$.*

Example 2.1.1. *Consider the nondeterministic finite state automaton (NFA), denoted by $N = (Q, \Sigma, \delta, Q_0)$ and shown in Fig. 2.1 (ignore, for now, the outputs – 0 or 1 – on each transition). In this particular example, we have state set $Q = \{q^{(1)}, q^{(2)}, q^{(3)}, q^{(4)}\}$, input or event set $\Sigma = \{\alpha, \beta, \gamma\}$, and $Q_0 = Q$ (i.e., all states are possible initial states). The relation δ is captured by the label on each of the arrows that connect the various pairs of states (for example, $(q^{(1)}, \alpha, q^{(2)}) \in \delta$ is indicated by the arrow with label α from state $q^{(1)}$ to state $q^{(2)}$).*

Fig. 2.2 shows the observer for system N, based on input information only and assuming that all input events are observable, i.e., $\Sigma_o = \Sigma = \{\alpha, \beta, \gamma\}$. For example, we start at initial state $\{q^{(1)}, q^{(2)}, q^{(3)}, q^{(4)}\}$ (representing the fact that any initial state is possible) and, depending on the input, we move to different states: if α is applied, the set of possible states is $\{q^{(1)}, q^{(2)}, q^{(3)}, q^{(4)}\}$ (we get to $q^{(1)}$ from $q^{(4)}$, to $q^{(2)}$ from $q^{(1)}$ and $q^{(3)}$, to $q^{(3)}$ from $q^{(3)}$, to $q^{(4)}$ from $q^{(2)}$ and $q^{(4)}$); similarly, if β is applied, the set of possible states is $\{q^{(1)}, q^{(2)}, q^{(3)}, q^{(4)}\}$ (we get to $q^{(1)}$ from $q^{(1)}$, to $q^{(2)}$ from $q^{(4)}$, to $q^{(3)}$ from $q^{(2)}$, to $q^{(4)}$ from $q^{(1)}$ and $q^{(3)}$); finally, if γ is applied, the set of possible states is $\{q^{(1)}, q^{(2)}, q^{(3)}\}$ (we get to $q^{(1)}$ from $q^{(2)}$ and $q^{(3)}$, to $q^{(2)}$ from $q^{(2)}$, to $q^{(3)}$ from $q^{(1)}$ and $q^{(4)}$). For each of the newly obtained states, we can continue in a similar fashion. For example, from state $\{q^{(1)}, q^{(2)}, q^{(3)}\}$,we have the following possibilities: if α is applied, the set of possible states is $\{q^{(2)}, q^{(3)}, q^{(4)}\}$ (we get to $q^{(2)}$ from $q^{(1)}$ and $q^{(3)}$, to $q^{(3)}$ from $q^{(2)}$, and to $q^{(4)}$ from $q^{(1)}$). If β is applied, the set of possible states is $\{q^{(1)}, q^{(3)}, q^{(4)}\}$ (we get to $q^{(1)}$ from $q^{(1)}$, to $q^{(3)}$ from $q^{(2)}$, and to $q^{(4)}$ from $q^{(1)}$ and $q^{(3)}$); if γ is applied, the set of possible states is $\{q^{(1)}, q^{(2)}, q^{(3)}\}$. We can continue from each new state in a similar fashion until no new states are obtained.

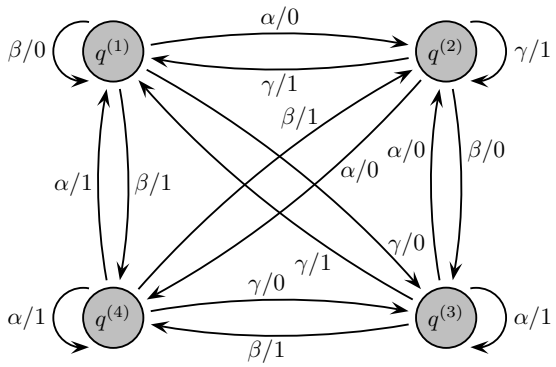

Figure 2.1: Nondeterministic finite automaton (NFA) with outputs.

Note that the dotted arrow in Fig. 2.2, represents the arrival at a singleton state that we are aiming for (thus, we do not consider continuations from such states). Also, to avoid cluttering the diagram, we color-code the states: observer states with the same color represent the same state (thus, we only describe continuations from one such state).

2.2 Fully Defined Nondeterministic Finite Automata with Outputs

In this work, we are interested in fully (or completely) defined nondeterministic finite automata (FDNFAs) with outputs, as described below.

Definition 2.2.1. *(Fully Defined Nondeterministic Finite Automata (FDNFA) with Outputs) A fully defined nondeterministic finite automaton (FDNFA) with outputs is captured by* $M = (Q, \Sigma, Y, \delta, \lambda, Q_0)$ *where* (Q, Σ, δ, Q_0) *is an NFA with* $Q = \{q^{(1)}, q^{(2)}, \ldots, q^{(|Q|)}\}$ *being the finite set of states,* Σ *being the finite set of inputs,* Q_0*,* $Q_0 \subseteq Q$*, being the set of possible initial states, and* δ *being a fully defined transition relation (i.e., for each* $q \in Q$ *and each* $\sigma \in \Sigma$*, we can find at least one* $q' \in Q$ *such that* $(q, \sigma, q') \in \delta$*). Moreover,* $Y = \{y^{(1)}, y^{(2)}, \ldots, y^{(|Y|)}\}$ *is the finite set of outputs and* $\lambda : Q \times \Sigma \times Q \to Y$ *is the output function that specifies, for each triplet* $(q, \sigma, q') \in \delta$*, the output* $y \in Y$ *that the system produces.*

The output function λ is assumed without loss of generality to be surjective. Given an FDNFA with starting state q_0, $q_0 \in Q_0$, if we apply a sequence of inputs $s \in \Sigma^*$, denoted by $s = \sigma_1 \sigma_2 \ldots \sigma_k$,

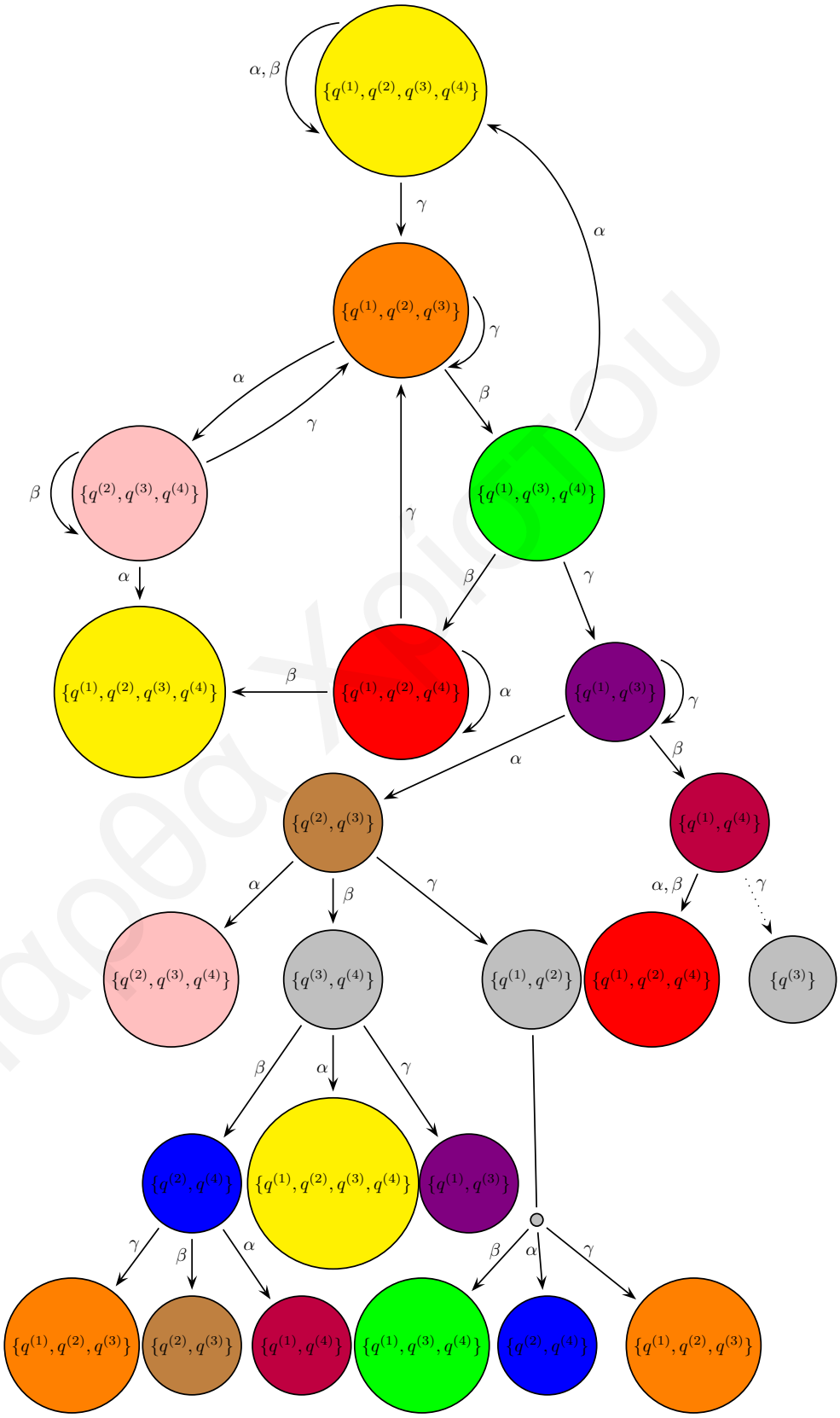

Figure 2.2: Construction of the observer for the NFA in Fig. 2.1 based on inputs only.

we can obtain a resulting sequence of states and outputs as follows:

$$\begin{array}{rcl} q_1 & \in & \delta(q_0,\sigma_1) \\ y_1 & = & \lambda(q_0,\sigma_1,q_1) \\ q_2 & \in & \delta(q_1,\sigma_2) \\ y_2 & = & \lambda(q_1,\sigma_2,q_2) \\ & \vdots & \\ q_k & \in & \delta(q_{k-1},\sigma_k) \\ y_k & = & \lambda(q_{k-1},\sigma_k,q_k)\,. \end{array}$$

Since the FDNFA is nondeterministic, multiple state sequences (and corresponding sequences of outputs) can be obtained. The next example illustrates the notation that is used.

Example 2.2.1. *Consider the FDNFA with outputs,* $M = (Q,\Sigma,Y,\delta,\lambda,Q_0)$ *shown in Fig. 2.1. In this particular example, we have state set* $Q = \{q^{(1)},q^{(2)},q^{(3)},q^{(4)}\}$*, input set* $\Sigma = \{\alpha,\beta,\gamma\}$*, output set* $Y = \{0,1\}$*, and* $Q_0 = Q$ *(i.e., all states are possible initial states). As mentioned earlier the relation* δ *is captured by the arrows in the figure: for example,* $(q^{(1)},\alpha,q^{(2)}) \in \delta$ *and this is indicated by the label* α *on the arrow from state* $q^{(1)}$ *to state* $q^{(2)}$*; similarly,* $(q^{(2)},\alpha,q^{(4)}) \in \delta$ *and this is indicated by the label* α *on the arrow from state* $q^{(2)}$ *to state* $q^{(4)}$*; and so forth.*

The function λ *is also captured by the labels on the arrows of the figure: for example,* $\lambda(q^{(1)},\alpha,q^{(2)}) = 0$ *and this is captured by the label* 0 *on the arrow from state* $q^{(1)}$ *to state* $q^{(2)}$ *with input* α*; similarly,* $\lambda(q^{(2)},\alpha,q^{(4)}) = 0$ *and this is captured by the label* 0 *on the arrow from state* $q^{(2)}$ *to state* $q^{(4)}$ *with label* α*; and so forth.*

In summary, the label "σ/y*" on an arrow from state* $q^{(i)}$ *to state* $q^{(j)}$ *indicates that* $(q^{(i)},\sigma,q^{(j)}) \in \delta$ *and* $\lambda(q^{(i)},\sigma,q^{(j)}) = y$*.*

If the sequence of inputs $\alpha\beta$ *is applied and the sequence of outputs* 00 *is observed, we have the following matching sequence of transitions:* $\alpha/0$*,* $\beta/0$*. In this particular example, this leads to two matching state trajectories, namely*

$$\begin{array}{l} q^{(1)},q^{(2)},q^{(3)}\,, \\ q^{(3)},q^{(2)},q^{(3)}\,, \end{array}$$

and implies that the set of possible initial states is $\{q^{(1)},q^{(3)}\}$ *and the set of possible current states is* $\{q^{(3)}\}$*.*

2.3 Strong Detectability and its Verification

We recall below the notion of strong detectability [8] which is of interest in this thesis.

Definition 2.3.1. *(Strong Detectability [8]) An NFA $N = (Q, \Sigma, \delta, Q_0)$ is strongly detectable with respect to natural projection map P for the set of observable events $\Sigma_o \subseteq \Sigma$ if for* all *system trajectories s, $s \in L(N)$, we can determine (exactly) the current state and subsequent states of the system, after a finite number of observations (greater than a certain critical length). In other words, $\exists n_c \in \mathbb{N}$ such that $\forall n \geq n_c$ we have*

$$(\forall s \in \Sigma^* : |P(s)| = n) \Rightarrow |R(Q_0, P(s))| \leq 1 \ .$$

Remark 2.3.1. *For all $s \in L(N)$, we have $|R(Q_0, P(s))| \geq 1$, thus in the above definition we could have required that*

$$(\forall s \in L(N) : |P(s)| = n) \Rightarrow |R(Q_0, P(s))| = 1 \ .$$

Strong detectability for NFA N can be verified easily by constructing its observer N_{obs} and checking whether it has loops with certain properties [8]. The observer (or current state estimator) of N was described in the beginning of this chapter and is a standard construction that captures the possible current states in NFA N following any sequence of observations.

Definition 2.3.2. *(Observer or Current-State Estimator [7]) Given an NFA $N = (Q, \Sigma, \delta, Q_0)$ with set of observable events $\Sigma_o \subseteq \Sigma$ under the natural projection map P, the observer (or current-state estimator) is a deterministic finite automaton (DFA) $N_{obs} = (Q_{obs}, \Sigma_o, \delta_{obs}, Q_{0,obs})$, which can be constructed as follows:*

1. *Each state of N_{obs} is associated with a unique subset of states of the original NFA N (this means that $Q_{obs} \subseteq 2^Q$ has at most $2^{|Q|}$ states).*
2. *The initial state $Q_{0,obs}$ of N_{obs} is the unobservable reach of Q_0 (i.e., $Q_{0,obs} = UR(Q_0) = R(Q_0, \varepsilon)$).*
3. *From any state $q_{obs} \in Q_{obs}$ of the current-state estimator, the next state for any $\sigma_o \in \Sigma_o$ is captured by $\delta_{obs}(q_{obs}, \sigma_o) = R(q_{obs}, \sigma_o)$.*

Theorem 2.3.1. *(Strong detectability: Necessary and sufficient conditions using observer N_{obs} [8]) An NFA $N = (Q, \Sigma, \delta, Q_0)$ is strongly detectable with respect to the natural projection P for a set of observable events Σ_o, $\Sigma_o \subseteq \Sigma$, iff its observer $N_{obs} = (Q_{obs}, \Sigma_o, \delta_{obs}, Q_{0,obs})$ does not include loops*

that contain ambiguous states, or are followed by an ambiguous state. Note that a state of N_{obs} is ambiguous *if it is associated with a set of states of N that has cardinality greater or equal to* 2.

Apart from verifying detectability using an observer (as captured by the conditions in Theorem 2.3.1), strong detectability for NFA N can also be verified with polynomial complexity using a detector.

Definition 2.3.3. *(Detector [9]) Given an NFA $N = (Q, \Sigma, \delta, Q_0)$ under the natural projection map P with respect to the set of observable events Σ_o, $\Sigma_o \subseteq \Sigma$, the detector $N_d = (Q_d, \Sigma_o, \delta_d, Q_{0,d})$ is a nondeterministic finite automaton, where*

1. *$Q_d = \{Q_{0,d}\} \cup Q_s \cup Q_p$ is the finite set of states, with*
   (i) $Q_{0,d} = R(Q_0, \varepsilon)$ being the set of all possible initial states for NFA N before any observation is made;
   (ii) $Q_s = \{\{q_j\} \mid q_j \in Q\}$; and
   (iii) $Q_p = \{q_{d_1}, q_{d_2}, ..., q_{d_D}\}$, where $D = |Q \times Q| - |Q|$ with $q_{d_i} = \{q_l, q_m\} \in Q_p$, $q_l \neq q_m$, $q_l, q_m \in Q$.
2. *$\delta_d : Q_d \times \Sigma_o \to Q_d$ captures the state transitions and is defined as follows:*

$$\delta_d(q_d, \sigma) = \begin{cases} \{q_{d_i} \in Q_p | q_{d_i} \subseteq R(q_d, \sigma)\}, & \text{if } |R(q_d, \sigma)| > 1, \\ \{q_l\} \in Q_s, & \text{if } R(q_d, \sigma) = \{q_l\}, \\ \text{undefined}, & \text{if } R(q_d, \sigma) = \emptyset. \end{cases}$$

Theorem 2.3.2. *(Strong detectability: Necessary and sufficient conditions using detector N_d [9]) NFA N is strongly detectable iff its detector N_d does not include any loop that is reachable from the initial state and contains ambiguous states, or is followed by ambiguous states (ambiguous state are states in Q_p).*

Remark 2.3.2. *The detector N_d (in Definition 2.3.3) can be used to verify strong detectability for an NFA N with polynomial complexity (with respect to the size of the given NFA). The reason is that the number of states of the detector is at most $|Q|^2 + 1$; this should be contrasted with the number of states of the observer which could be as high as $2^{|Q|}$.*

2.4 K-Detectability and its Verification

In this section we define the notion of strong K-detectability (one can similarly define K-detectability, strong periodic K-detectability, and periodic K-detectability, see [8]).

Definition 2.4.1. *(Strong K-Detectability) Given a positive integer K, an NFA $N = (Q, \Sigma, \delta, Q_0)$ is strongly K-detectable with respect to the natural projection map P for the set of observable events $\Sigma_o \subseteq \Sigma$ if for* all *system trajectories s, $s \in L(N)$, we can determine, within a subset of states of cardinality no more than K, the current state and subsequent states of the system, after a finite number of observations (greater than a certain critical length n_c). Mathematically, $\exists n_c \in \mathbb{N}$ such that $\forall n \geq n_c$ we have*

$$(\forall s \in \Sigma^* : |P(s)| = n) \Rightarrow |R(Q_0, P(s))| \leq K \ .$$

Remark 2.4.1. *Note that in [18] a different notion of K-detectability is used to capture the ability to detect the precise current-state of the system (without uncertainty) after K steps; perhaps a better terminology for the notion in [18] would be K-step detectability.*

Clearly, strong K-detectability for NFA N can be verified easily by constructing its observer N_{obs} and checking whether it has loops with certain properties. In particular, the observer should not have loops that involve sets of state estimates of cardinality greater than K.

Theorem 2.4.1. *(Strong K-detectability: Necessary and sufficient conditions using observer N_{obs}) An NFA $N = (Q, \Sigma, \delta, Q_0)$ is strongly K-detectable with respect to the natural projection P for a set of observable events Σ_o, $\Sigma_o \subseteq \Sigma$, iff its observer $N_{obs} = (Q_{obs}, \Sigma_o, \delta_{obs}, Q_{0,obs})$ does not include loops that contain or are followed by states associated with sets of state estimates of cardinality greater than K.*

We next argue that, apart from verifying K-detectability using an observer, strong K-detectability for NFA N can also be verified with polynomial complexity using a K-detector. One can view the K-detector as a construction that aims to resemble the observer for states for which the associated sets of state estimates have cardinality K or less, whereas it resembles the detector for states for which the associated sets of state estimates have cardinality $K+1$ or more.

Definition 2.4.2. *(K-Detector) Given an NFA $N = (Q, \Sigma, \delta, Q_0)$ under the natural projection map P with respect to the set of observable events Σ_o, $\Sigma_o \subseteq \Sigma$, the K-detector $N_{kd} = (Q_{kd}, \Sigma_o, \delta_{kd}, Q_{0,kd})$ is a nondeterministic finite automaton, where*

1. *$Q_{kd} = \{Q_{0,kd}\} \cup Q_s \cup Q_p$ is the finite set of states, with*
   (i) $Q_{0,kd} = R(Q_0, \varepsilon)$ being the set of all possible initial states for NFA N before any observation is made;
   (ii) $Q_s = \{S_s \mid S_s \subseteq Q \wedge |S_s| \leq K\}$; and

(iii) $Q_p = \{S_p \mid S_p \subseteq Q \wedge |S_p| = K+1\}$.

2. $\delta_{kd} : Q_{kd} \times \Sigma_o \to Q_{kd}$ *captures the state transitions and is defined as follows:*

$$\delta_{kd}(q_{kd}, \sigma) = \begin{cases} \{S_s \in Q_s \mid S_s = R(q_{kd}, \sigma)\}, & \text{if } |R(q_{kd}, \sigma)| \leq K, \\ \{S_p \in Q_p \mid S_p \subseteq R(q_{kd}, \sigma)\}, & \text{if } |R(q_{kd}, \sigma)| > K. \end{cases}$$

Remark 2.4.2. *For simplicity of notation, the K-detector has a trapping state that corresponds to the empty set of current state estimates. This state was not included in the construction of the standard detector in Definition 2.3.3, but we could have easily done it by requiring all transitions that were left undefined to lead to this trapping state. Apart from this difference, the K-detector defined above reduces for* $K = 1$ *to the standard detector in Definition 2.3.3.*

Theorem 2.4.2. *(Strong K-detectability: Necessary and sufficient conditions using the K-detector* N_{kd} *[9]) NFA N is strongly K-detectable iff its K-detector* N_{kd} *does not include any loop that is reachable from the initial state and contains ambiguous states or is followed by ambiguous states (ambiguous states in the case of the K-detector are the states in* Q_p*).*

Remark 2.4.3. *The structure of the K-detector within the set* Q_s *is not necessarily identical to the structure of the observer for states that are associated with sets of state estimates of cardinality K or less. However, it should also be clear from Theorem 2.4.2 that the structure of the observer within the set* Q_s *is replicated by the K-detector (but the K-detector could have additional states).*

Remark 2.4.4. *Given an NFA N with n states, its K-detector* N_{kd} *(in Definition 2.4.2) is a construction that in the worst case has the following state complexity:*
(i) $2^K = O(n^K)$ *states in the set* Q_s*;*
(ii) $\begin{pmatrix} n \\ K+1 \end{pmatrix} = O(n^K)$ *states in the set* Q_p*;*
(iii) possibly one additional state as the initial state $X_{0,kd}$*.*
Thus, the K-detector has a total of $O(n^K)$ *states (the number of transitions per state could be in the worst case* $|\Sigma_o|$ *for a total number of* $O(|\Sigma_o|n^K)$ *transitions. Therefore, using the K-detector, the verification of strong K-detectability for an NFA N can be accomplished with complexity that is polynomial in the number of states of N and exponential in K.*

Chapter 3

Problem Formulation and Solution Approach

The problem we are interested in solving is the following: given a fully defined nondeterministic finite automaton (FDNFA) $N = (Q, \Sigma, Y, \delta, \lambda, Q_0)$, we need to apply a sequence of inputs $\sigma[1], \sigma[2], \ldots, \sigma[n]$, such that, once this sequence is applied, the sequence of outputs $y[1], y[2], \ldots, y[n]$ that is generated allows us to identify exactly the final (current) state of the system. The input sequence could be fixed (determined ahead of time) or adaptive ($\sigma[k]$, $k = 2 \ldots, n$, is determined based on the previous inputs $\sigma[1], \sigma[2], \ldots, \sigma[k-1]$ and the previous outputs $y[1], y[2], \ldots, y[k-1]$). In this thesis, we focus on the latter case, i.e., at each time epoch of the system operation, we can decide which input to apply, based on the outputs that have been observed thus far (as well as the sequence of inputs that we have applied). The goal is to choose the sequence of inputs so that eventually we know the state of the system exactly.

Given an FDNFA as defined in the previous section, we can construct an observer-like structure (refer to Fig. 3.1), based on the inputs, which are chosen, and the outputs, which are produced ("chosen") by the system. In Fig. 3.1 inputs are drawn using the shape of a circle, whereas outputs are depicted with the shape of a square. In other words, we separate inputs and outputs in a way that allows us to easily recognize them in a diagram: circles correspond to controllable events whereas squares correspond to uncontrollable events. The advantage of the resulting structure is that it summarizes all possible state estimates based on inputs and outputs.

There are several ways to approach the solution for the problem described at the beginning of this section. An interesting approach is inspired from controlled detectability [11] where one attempts

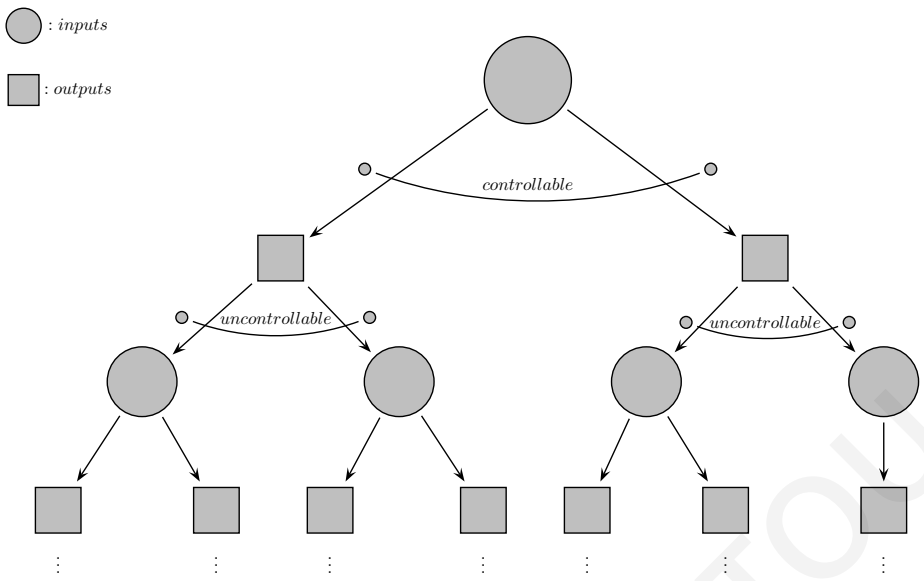

Figure 3.1: Problem formulation

to control the system input sequence so as to make the system detectable. This is necessary or desirable in situations where there is some behavior in the system that makes its state undetectable [11]. Our problem can also be considered as a supervisory control strategy for avoiding behavior that keeps the state of the system undetectable. The natural question that arises is how to obtain this supervisory control strategy systematically.

Our goal is to obtain a control strategy that allows us to reach a singleton state in the observer construction. Note that a singleton observer state is a state associated with a set of possible states that has cardinality exactly one (e.g., $\{q^{(1)}\}$). This goal does not mean that all the non-singleton states must be avoided since we could visit non-singleton states in route to a singleton state. In the next example, we illustrate the construction of the observer for the case of an FDNFA with outputs. This observer maintains the set of possible states by separately tracking both the input that is applied and the output that is produced.

Example 3.0.1. *In Fig. 3.2, we show part of the observer construction, based on inputs and outputs, for the FDNFA with outputs in Fig. 2.1. We assume that the set of initial states is* $Q_0 = Q = \{q^{(1)}, q^{(2)}, q^{(3)}, q^{(4)}\}$. *We start from initial state* Q_0 *at which point three possible inputs could be applied:* α, β, *and* γ. *We consider each case separately below.*

- *If* α *is applied, there are two possible output observations,* 0 *and* 1*: (i) if* 0 *is observed, then*

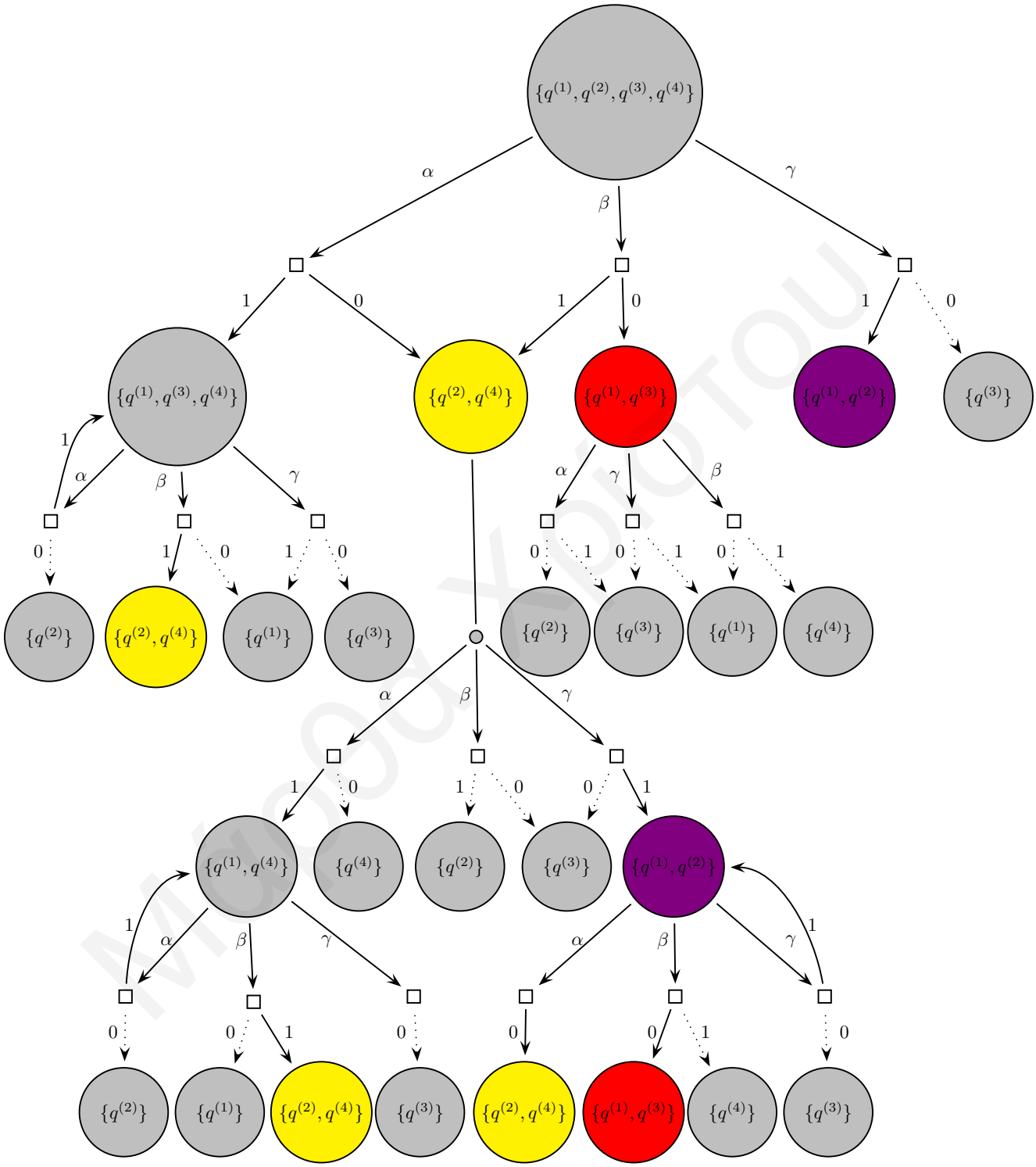

Figure 3.2: Observer construction based on inputs and outputs.

the set of possible states is $\{q^{(2)}, q^{(4)}\}$ *(*$q^{(2)}$ *can be reached via* $\alpha/0$ *from states* $q^{(1)}$ *and* $q^{(3)}$*, whereas* $q^{(4)}$ *can be reached via* $\alpha/0$ *from* $q^{(2)}$*); (ii) if* 1 *is observed, then the set of possible states is* $\{q^{(1)}, q^{(3)}, q^{(4)}\}$ *(*$q^{(1)}$ *can be reached via* $\alpha/1$ *from state* $q^{(4)}$*,* $q^{(3)}$ *can be reached via* $\alpha/1$ *from state* $q^{(3)}$*, and* $q^{(4)}$ *can be reached via* $\alpha/1$ *from state* $q^{(4)}$*).*

- *If* β *is applied, there are two possible output observations,* 0 *and* 1*: (i) if* 0 *is observed, then the set of possible states is* $\{q^{(1)}, q^{(3)}\}$ *(*$q^{(1)}$ *can be reached via* $\beta/0$ *from state* $q^{(1)}$*, and* $q^{(3)}$ *can be reached via* $\beta/0$ *from state* $q^{(2)}$*); (ii) if* 1 *is observed, then the set of possible states is* $\{q^{(2)}, q^{(4)}\}$ *(*$q^{(2)}$ *can be reached via* $\beta/1$ *from state* $q^{(4)}$*, whereas* $q^{(4)}$ *can be reached via* $\beta/1$ *from states* $q^{(1)}$ *and* $q^{(3)}$*).*

- *If* γ *is applied, there are two possible output observations,* 0 *and* 1*: (i) if* 0 *is observed, then the set of possible states is* $\{q^{(3)}\}$ *(*$q^{(3)}$ *can be reached via* $\gamma/0$ *from states* $q^{(1)}$ *and* $q^{(4)}$*); if 1 is observed, then the set of possible states is* $\{q^{(1)}, q^{(2)}\}$ *(*$q^{(1)}$ *can be reached via* $\gamma/1$ *from states* $q^{(2)}$ *and* $q^{(3)}$*, whereas* $q^{(2)}$ *can be reached via* $\gamma/1$ *from state* $q^{(2)}$*).*

We can continue this construction in the same manner, by considering each of the possible inputs and corresponding outputs, and the sets of states they lead to. For better readability, we do not consider continuations from states that are associated with singleton sets of state estimates (since the goal in Fig. 3.2 is to determine exactly the current state of the system). Also, the construction in Fig. 3.2 does not merge identical states but uses color coding instead to avoid cluttering the diagram (e.g., the state reached after the observation of $\alpha/0$ *is the same as the state reached after the observation of* $\alpha/1$ *followed by* $\beta/1$*).*

Once we complete the construction with inputs and outputs (part of which is shown in Fig. 3.2), we can use the graph structure to obtain possible strategies to solve the problem outlined in the beginning of this section (also refer to Fig. 3.5). Note that events associated with circles are controllable, whereas events associated with squares are uncontrollable. We find a solution with the following iteration: initially, we mark with a $\surd$ (which denotes desirable or acceptable states) all states that are singleton sets. Then, we iteratively mark states with a $\surd$ as follows:

- RULE 1: We put $\surd$ to a circle, if it has at least one subsequent square marked with a $\surd$ (since in circle states events are controllable (this rule is illustrated in Fig. 3.3).

- RULE 2: We put $\surd$ at a square state only when all subsequent circles are marked with $\surd$ (as

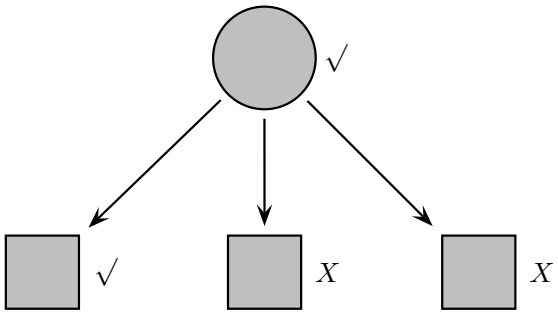

Figure 3.3: Illustration of Rule 1.

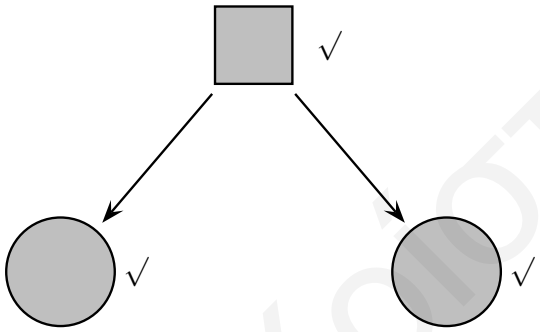

Figure 3.4: llustration of Rule 2.

illustrated in Fig. 3.4).

Fig. 3.5 shows the application of the above rules for the system in Fig. 2.1.

In summary, the strategy is as follows: we first mark all states associated with singleton sets as desirable states, by putting $\surd$. Then, we iteratively apply the two rules that we mentioned above from bottom to top. At the end of the iterative process, the following holds true: if the starting state is marked with $\surd$, then we have a strategy.

Remark 3.0.1. *Note that the maximum possible number of states for the observer in Fig. 3.2 is* $2^{|Q|}$. *In this example, we could have a maximum of* 16 *observer states; in reality, we have significantly less states.*

We next provide a pseudo code description of the proposed approach.

- Step1: Create an observer based on inputs and outputs.
- Step2: Select observer states associated with singleton sets and mark them with $\surd$.

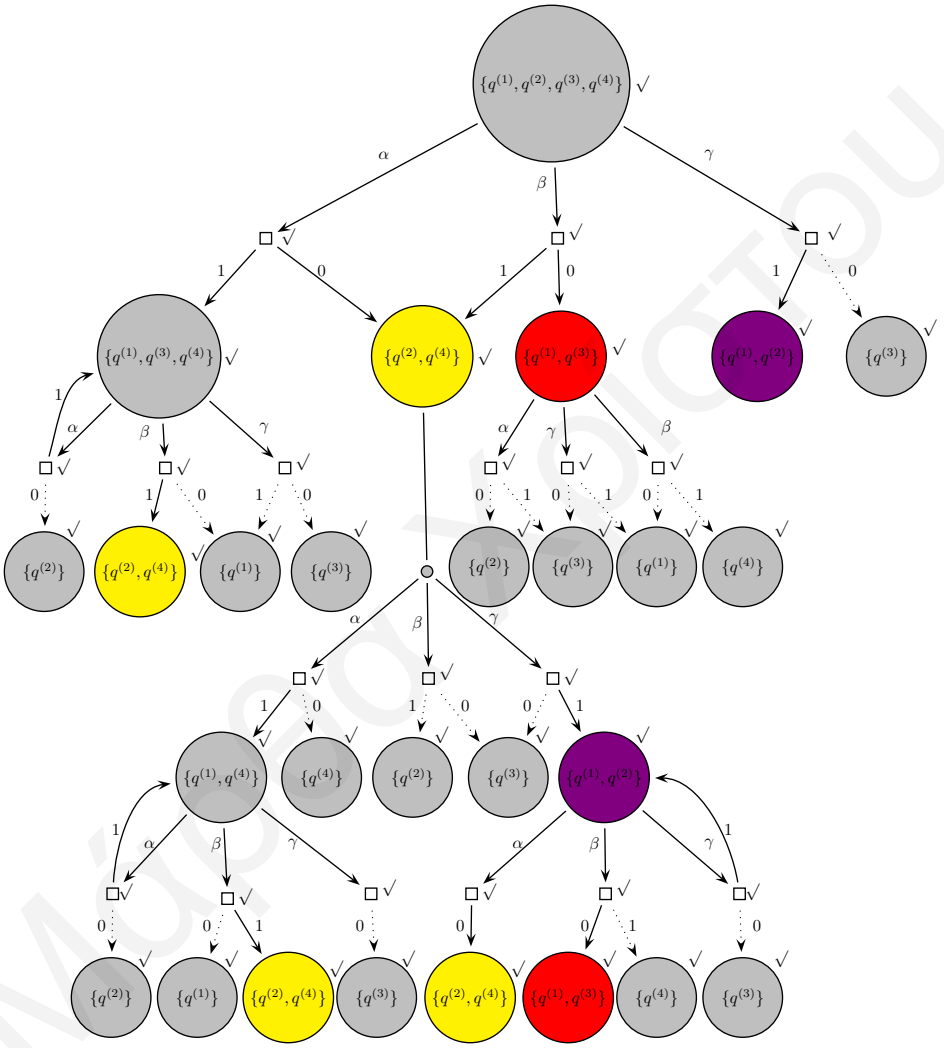

Figure 3.5: Solution approach.

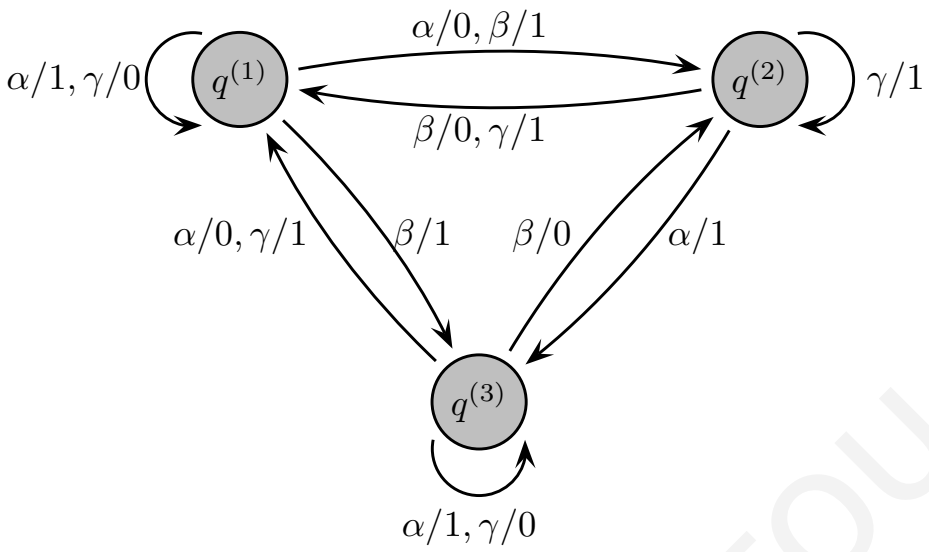

Figure 3.6: Nondeterministic finite automaton (NFA) with outputs.

- Step3: Iteratively apply Rule 1 and Rule 2 to the observer states starting from singleton states and subsequently considering precursor states.
- Step4: If the initial state is marked with $\surd$ then a solution strategy exists.

Example 3.0.2. *Consider the FDNFA with outputs, denoted by* $M = (Q, \Sigma, Y, \delta, \lambda, Q_0)$ *and shown in Fig. 3.6. In this particular example, we have state set* $Q = \{q^{(1)}, q^{(2)}, q^{(3)}\}$*, input set* $\Sigma = \{\alpha, \beta, \gamma\}$*, output set* $Y = \{0, 1\}$*, and* $Q_0 = Q$ *(i.e., all states are possible initial states). The functions* δ *and* λ *are as defined by the arrows and the labels in figure.*

The observer with inputs and outputs for the above system is shown in Fig. 3.7 it starts from a state that represents the set of initial states $Q_0 = Q = \{q^{(1)}, q^{(2)}, q^{(3)}\}$*; from this initial state, there are three possible inputs:* α*,* β*, and* γ*. We consider each case separately below.*

- *If* α *is applied, there are two possible output observations,* 0 *and* 1*:*
  (i) If 0 *is observed, then the set of possible states is* $\{q^{(1)}, q^{(2)}\}$ *(*$q^{(1)}$ *can be reached via* $\alpha/0$ *from state* $q^{(3)}$*, and* $q^{(2)}$ *can be reached via* $\alpha/0$ *from state* $q^{(1)}$*).*
  (ii) If 1 *is observed, then the set of possible states is* $\{q^{(1)}, q^{(3)}\}$ *(*$q^{(1)}$ *can be reached via* $\alpha/1$ *from state* $q^{(1)}$*, and* $q^{(3)}$ *can be reached via* $\alpha/1$ *from states* $q^{(2)}$ *and* $q^{(3)}$*).*
- *If* β *is applied, there are two possible output observations,* 0 *and* 1*:*

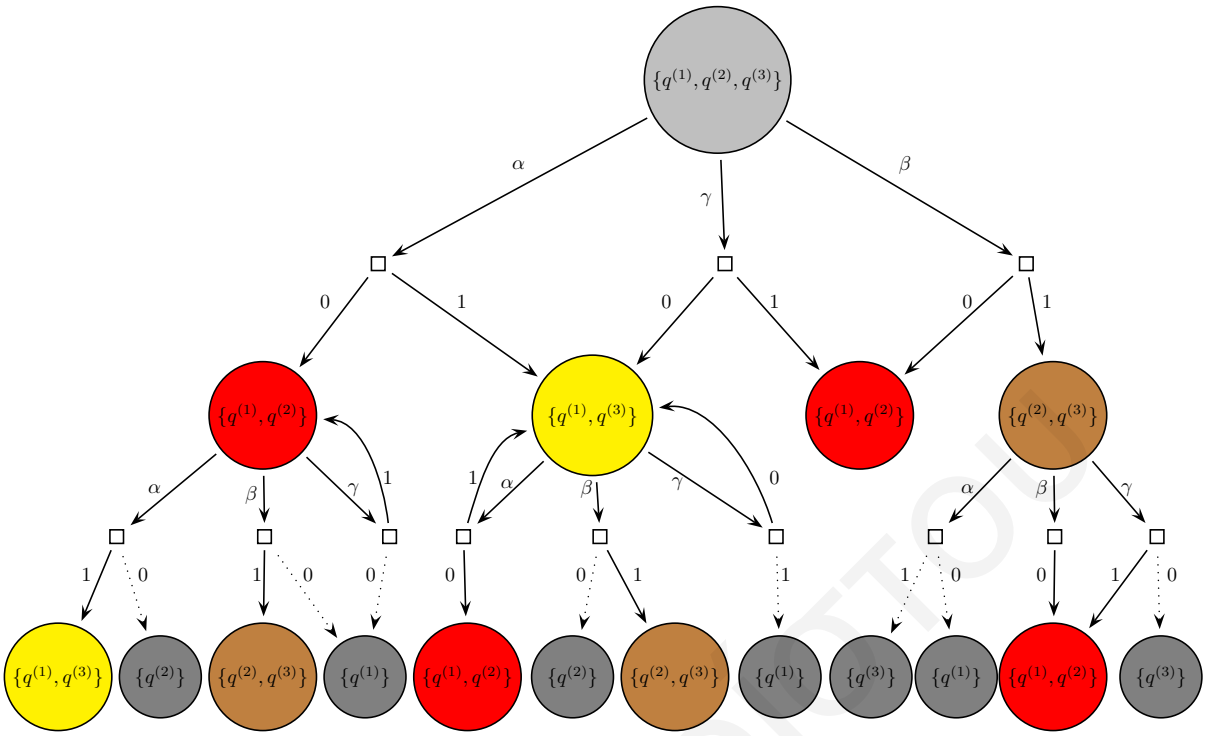

Figure 3.7: Construction of the observer with inputs and outputs for the NFA in Fig. 3.6.

(i) If 0 *is observed, then the set of possible states is* $\{q^{(1)}, q^{(2)}\}$ *(*$q^{(1)}$ *can be reached via* $\beta/0$ *from state* $q^{(2)}$*, and* $q^{(2)}$ *can be reached via* $\beta/0$ *from state* $q^{(3)}$*).*
(ii) If 1 *is observed, then the set of possible states is* $\{q^{(2)}, q^{(3)}\}$ *(*$q^{(2)}$ *can be reached via* $\beta/1$ *from state* $q^{(1)}$*, and* $q^{(3)}$ *can be reached via* $\beta/1$ *from state* $q^{(1)}$*).*

- *If* γ *is applied, there are two possible output observations,* 0 *and* 1*:*
  (i) If 0 *is observed, then the set of possible states is* $\{q^{(1)}, q^{(3)}\}$ *(*$q^{(1)}$ *can be reached via* $\gamma/0$ *from state* $q^{(1)}$*, and* $q^{(3)}$ *can be reached via* $\gamma/0$ *from state* $q^{(3)}$*).*
  (ii) If 1 *is observed, then the set of possible states is* $\{q^{(1)}, q^{(2)}\}$ *(*$q^{(1)}$ *can be reached via* $\gamma/1$ *from states* $q^{(2)}$ *and* $q^{(3)}$*, and* $q^{(2)}$ *can be reached via* $\gamma/1$ *from state* $q^{(2)}$*).*

Sometimes we choose to redraw a node that already exists in the graph so that we do not clutter the digraph; in such case, we use the same color for the two node. Nodes that are associated with singleton sets are also replicated for convenience; for these nodes we use gray color and do not consider continuations. Once we construct the observer with inputs and outputs for the given FDNFA, we can use the iterative procedure to determine whether there exists a strategy for getting to know the exact current state of the FDNFA by applying an adaptive input sequence. The result

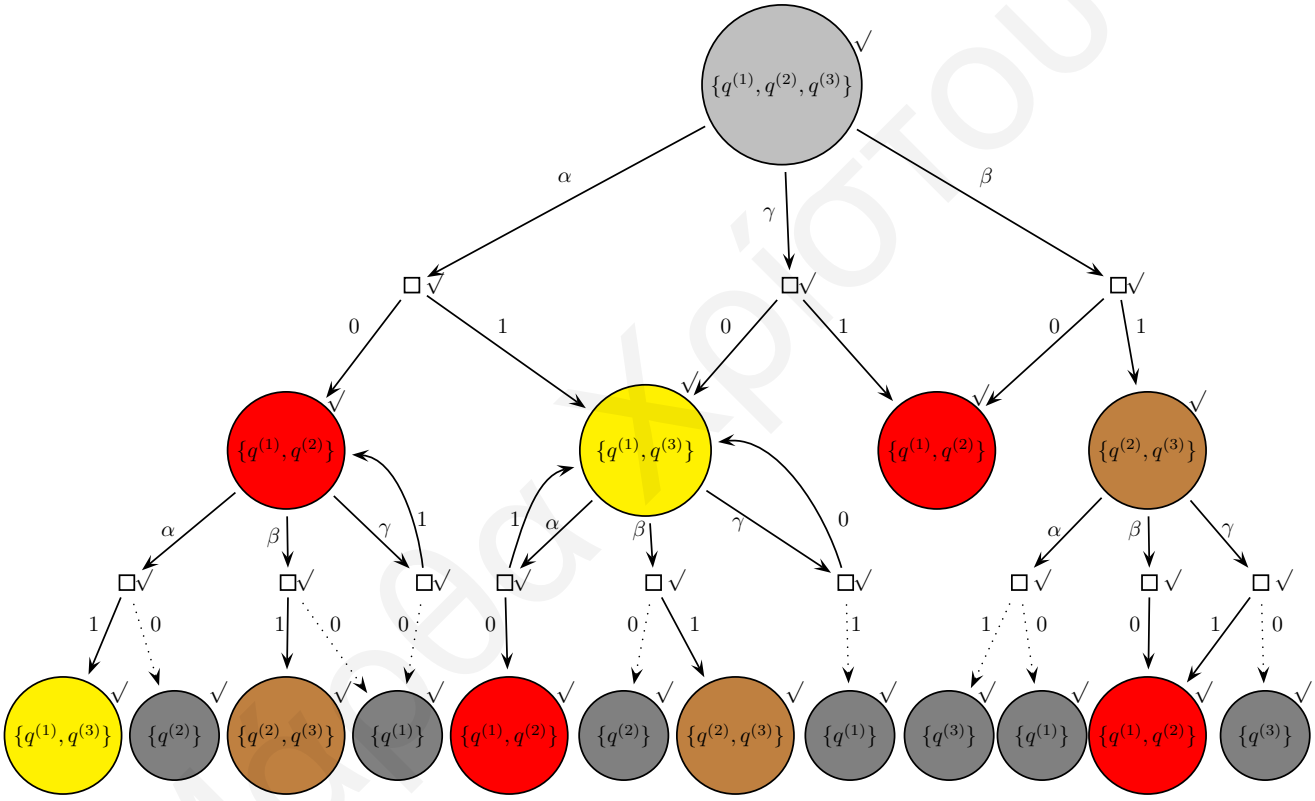

Figure 3.8: Solution approach for NFA with outputs in Fig. 3.6

of this procedure is shown in Fig. 3.8.

At initialization the only nodes in Fig. 3.8 that take a $\surd$ are the singleton subsets. After that, as shown in the figure, the only "square" nodes that could take a $\surd$ are the ones for which both of the leafs are singleton subset. Later on, with the application of the two rules, the state with brown color (i.e., $\{q^{(2)}, q^{(3)}\}$) takes a $\surd$, because it has at least one choice that drive us into a singleton subset. When the brown state takes a $\surd$, the only square node that should take a $\surd$ is the one reached from $\{q^{(1)}, q^{(2)}\}$ (because it has two children with a $\surd$), and the square node reached from $\{q^{(1)}, q^{(3)}\}$ with β (for the same reason). Continuing in this way, we can mark states iteratively. Following the above process, if the initial node gets marked with a $\surd$, then there at least one adaptive strategy that takes us into a singleton subset.

Note that if a square node or a circle node has a $\surd$, it means that there exists an adaptive sequence of subsequent decisions that ensures that the state of the system will be known exactly. A $\surd$ in a state does not mean that any choice from such states will eventually allow us to know the exact state of the system. The problem is that states with $\surd$ could form cycles; thus, repetitive careless choices (along with specific outputs provided by the system) could prevent us from reaching a situation where the state of the system is known exactly. Nevertheless, careful choices can always take us out of such cycles of states with uncertain state estimates.

Chapter 4

Min-Max Strategies

Following the algorithm of the previous section, we can determine whether there exists a strategy for eventually determining the current state of the system based on the status of the starting node (i.e., whether the starting node eventually gets marked with a $\surd$).

In general, at the end of the algorithm, there is a possibility for more than one strategies. In fact, when we have multiple options to choose from, any one of them would give satisfactory results. The question of which strategy is the best depends on the optimization criterion. One criterion for choosing the "best" strategy is based on the number of steps (inputs) that will be needed before the state of the system becomes exactly known. One should keep in mind that this number also depends on the actions of the system: some system actions may result in fewer number of steps than others. For this reason, a commonly adopted optimization criterion in such type of problems is to choose the strategy that results in the smallest number of steps, under the worst case scenario (with respect to the actions of the system).

From Fig. 3.5 (which is the result of the iterative application of Rules 1 and 2), we would like to obtain a structure like the one in Fig. 4.1, i.e., a tree that includes all possible strategies after the application of the algorithm. For example, from $\{q^{(1)}, q^{(4)}\}$ we remove input α (it is not an option because we might get 1); from the set of states $\{q^{(1)}, q^{(2)}\}$ we remove the input γ (it is not an option because we might get 1); and from $\{q^{(1)}, q^{(3)}, q^{(4)}\}$ we remove the input α (it is not an option because we might get 1).

Note that the structure in Fig. 4.1 summarizes all allowable strategies. At the starting state, we can choose, for example, α; then if we get 1 we will choose γ and we are done; however, if we get

$\underline{ROUNDS}$:

Round 1 : *Singleton States with* $C = 0$, $Non - Singleton$ *with* $C = \infty$.
Round 2 : *Update the Costs with* **C**
Round 3 : *Update the Costs with* **C**
Round 4 : *Update the Costs with* **C**
Round 5 : *Update the Costs with* **C**

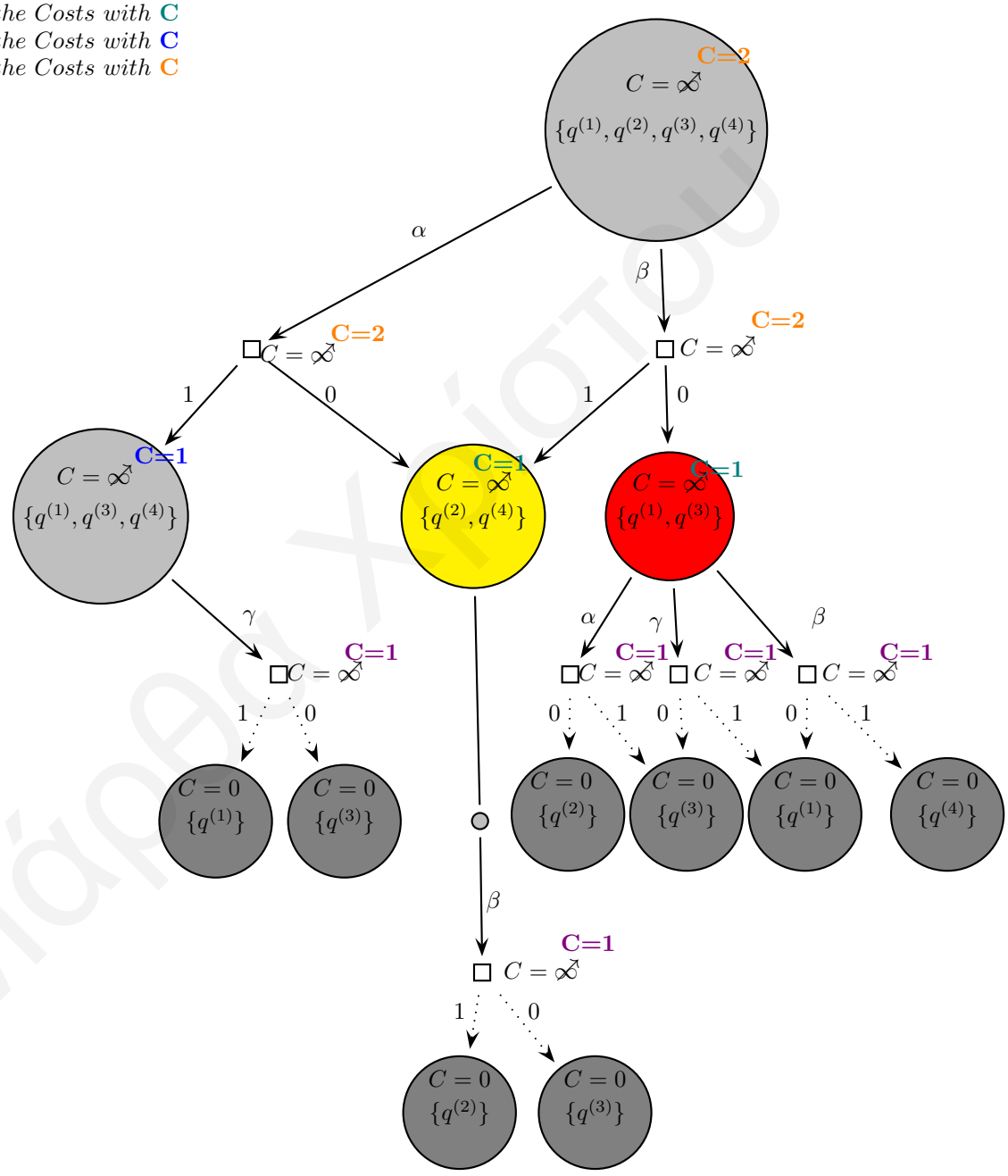

Figure 4.1: Acyclic Digraph.

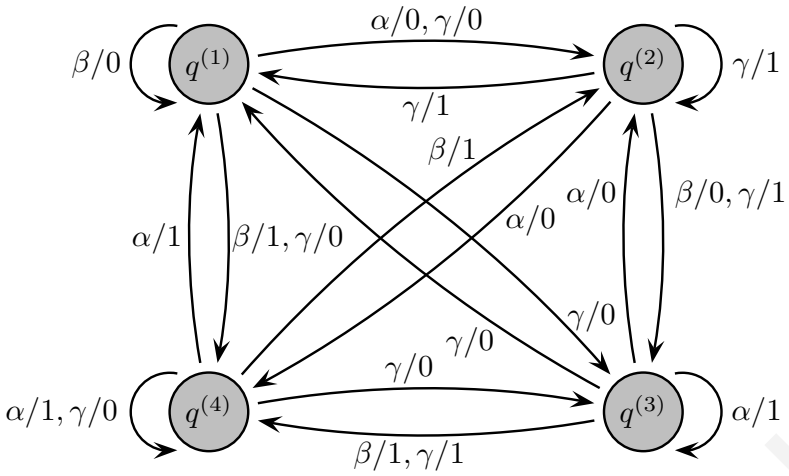

Figure 4.2: FDNFA considered in Example 2.

0, we will have to choose β and then we will be done (in either case, we get done at two steps). Suppose we choose β in the beginning. If we get 1, we will choose β and we will be done; if we get 0, we can choose anything and we will be done (again in two steps).

We now focus on how to ensure that the strategy we have selected will result in the minimum number of steps. The idea is that, at every state we can determine our strategy by choosing the input based on what is the *biggest/worst* height of *any* subsequent path at the tree. If we have this information for each possible subsequent input choice, we can choose the input (or inputs) that guarantees (or guarantee) the minimum (worst case) subsequent path. The pseudo code for the min-max strategy is described below.

Pseudocode: Min/Max Strategy **Algorithm 2:** Min/Max Iteration for Optimal Strategy

- Step 1: Create an observer with inputs and outputs with all possible choices that we have when the system runs.
- Step 2: Assign a cost variable c_q to each observer state (circle node) or square node q of the observer with input and outputs. Initialize this cost variable to $+\infty$, except at observer states (circle nodes) that are associated with singleton sets of state estimates, which receive an initial cost of zero (refer to Fig. 4.1).
- Step 3: Iteratively perform the following (until no more node costs get updated):
  1) Each square node q computes the maximum cost $c_{\max}$ among its subsequent nodes (circle nodes that can be reached from square node q). It then updates its cost c_q to be the minimum

of its current cost and $1 + c_{\max}$, i.e.,

$$c_q := \min(c_q, 1 + c_{\max}) ,$$

where

$$c_{\max} = \max_{q' \in out(q)} c_{q'}$$

with $out(q)$ being the set of observer states (circle nodes) that can be reached from q.

2) Each circle node q computes the minimum cost $c_{\min}$ among its subsequent nodes (square nodes that can be reached from circle node q). It then updates its cost c_q to be the minimum of its current cost and $c_{\min}$, i.e.,

$$c_q := \min(c_q, c_{\min}) ,$$

where

$$c_{\min} = \min_{q' \in out(q)} c_{q'}$$

with $out(q)$ being the set of square nodes that can be reached from q in the observer construction.

When we run the above iteration, we end up with a cost $c_{Q_{0,obs}}$ at the starting node of the observer. This cost corresponds to the smallest number of inputs that need to be applied in order to ensure that we reach a singleton state, regardless of the output sequence generated by the system. To obtain an optimal minimax strategy, at each circle node q, we only allow input choices that lead to square nodes with costs equal to c_q (the remaining inputs have higher cost which means that, in the worst case, they might require longer sequences of inputs before they lead to an observer state with state estimates that form a singleton set).

We next consider a modified example where we introduce additional uncertainty to input γ as in Fig. 4.3. We apply the algorithm that we mentioned above, to find if there exists an appropriate solution and at what cost. The corresponding observer with inputs and ouptus is shown in Fig. 4.4. When we apply the min-max algorithm, we conclude that "the best option" is to first apply input β.

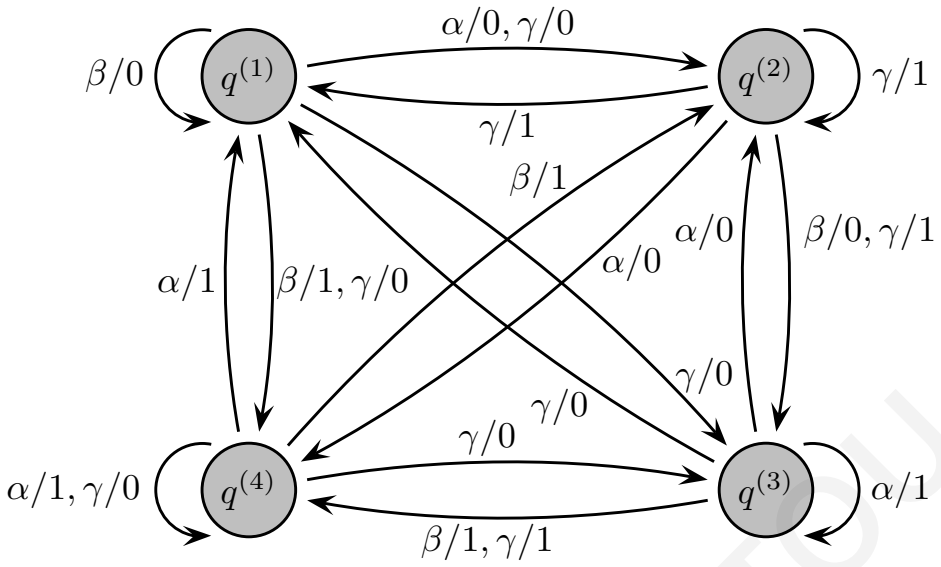

Figure 4.3: FDNFA considered in Example 2.

By applying the first algorithm (i.e., Rule 1 and Rule 2), we notice that the original state has a $\surd$, which means that there is at least one strategy that is guaranteed to lead us to a singleton set.

Following the iteration for the min-max strategy the cost at the root node is finite if and only if there is an adaptive strategy. Also, the cost at the root is the smallest number of the inputs that need to be applied and guarantee that the state of the system will be known exactly, regardless of which action the system takes.

To find the finite cost of the root node, we initialize the cost of each element (i.e., the cost of states associated with singleton sets is set to C=0 and the cost of the non-singleton sets and 'square' nodes are set to $C = \infty$). Afterwards, in subsequent iterations, we apply the algorithm for the min-max strategy. At each round, we update first the "square" nodes (one plus max of the cost of the leaves) and then the 'circle" nodes (min of the cost of the leaves).

Specifically, at iteration we update the cost of the "square" node which is connected with two singleton sets (i.e., max cost is C=0). This node receives a cost of 1 (as mentioned before on the cost equation we should add plus one to the max cost). Thus, the new cost of the "square" node is C=1. Moreover, the cost of the 'brown round' node will also change. It will become the min cost among all "square" nodes connected to it. We follow the same procedure until the cost of all nodes stop getting updated.

With the min-max algorithm, we want to obtain the minimum cost that will be necessary to find

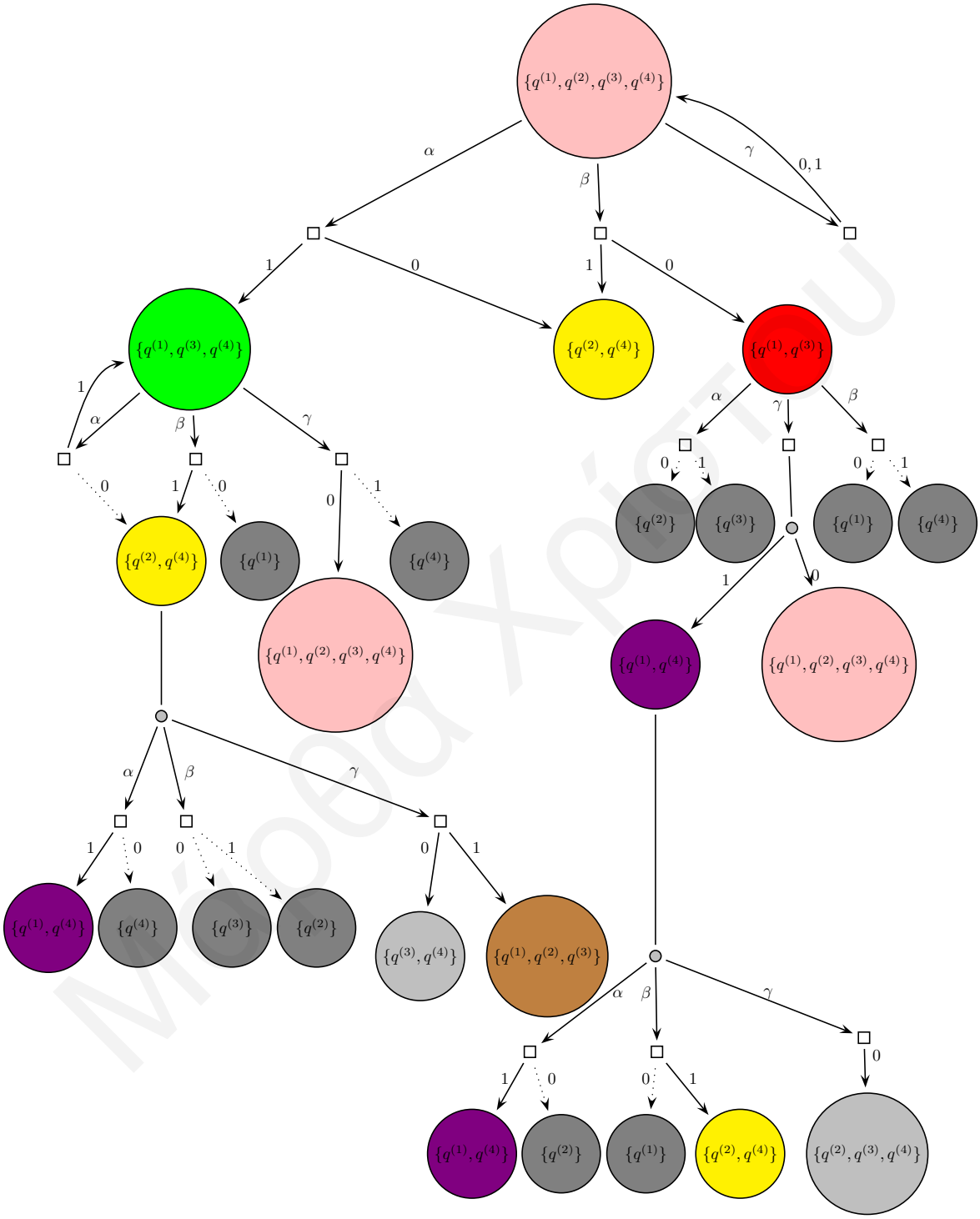

Figure 4.4: Observer construction for the system in Fig. 4.3

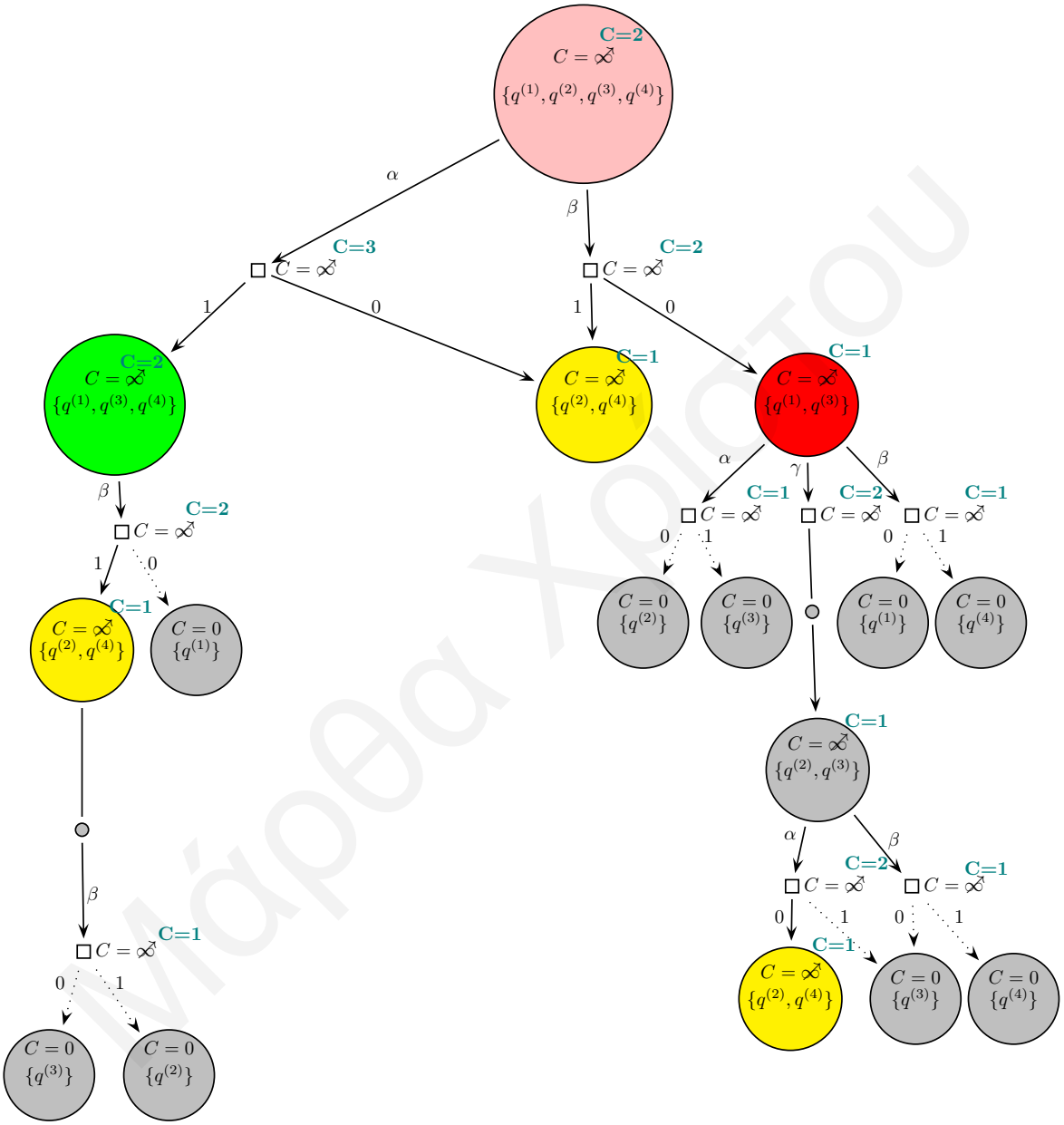

Figure 4.5: Feasible Strategies for the system in Fig. 4.3.

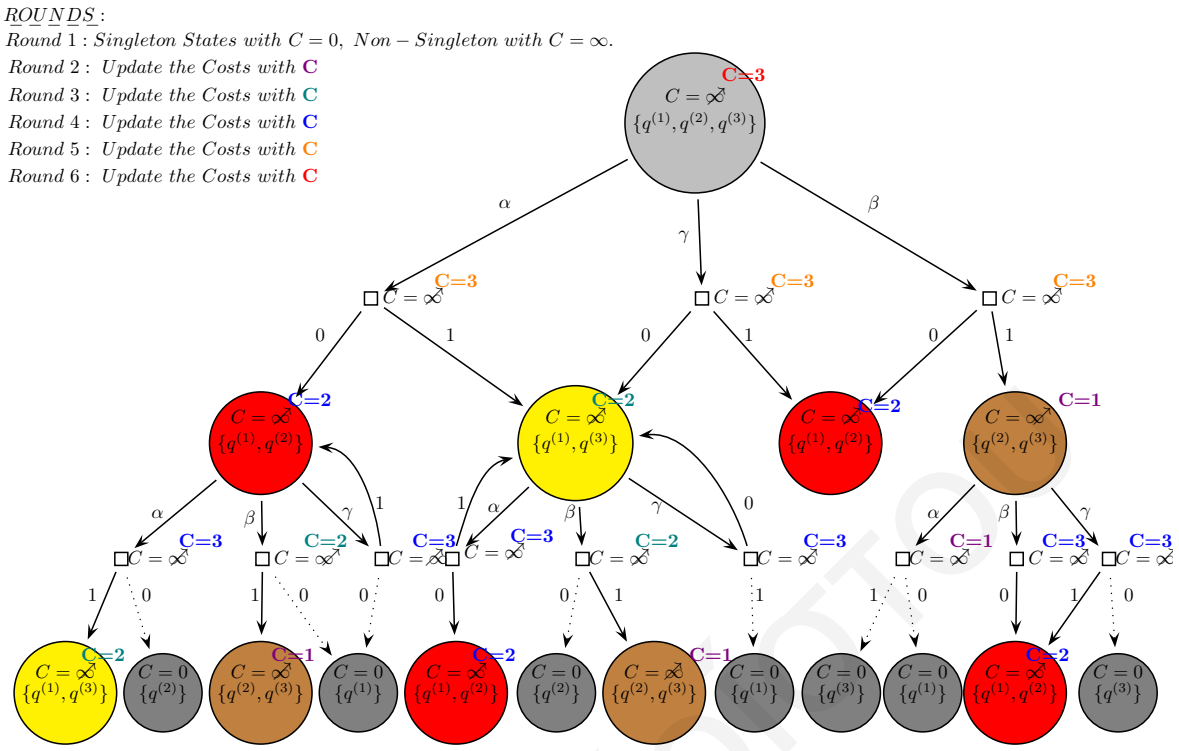

Figure 4.6: Cost observer of NDNFA Fig. 3.6.

the "best" solution. If the initial state of the observer has finite cost, this means a solution exists.

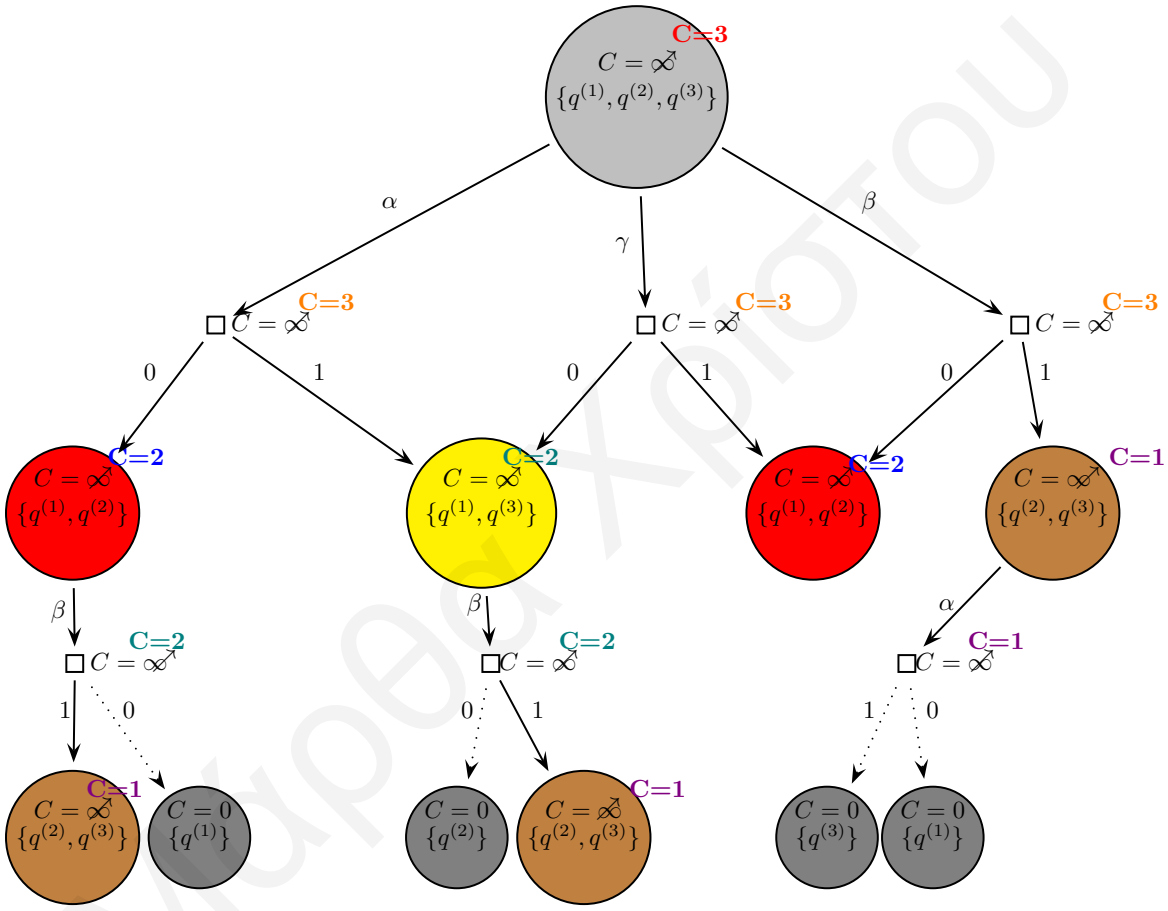

Figure 4.7: Possible strategies for the system in Fig. 3.7.

Chapter 5

Conclusions

5.1 Summary

This thesis focused on adaptive control strategies that choose the input based on the output produced by the system. The formulation of the problem assumes that we can control all inputs and, depending on the output, determines whether there exist a solution and which strategy is the "best" according to a min-max criterion.

We obtained and analysed adaptive control strategies. When we use adaptive strategies, we can carefully choose inputs, based on which the system generates outputs. Our model is that of a fully defined non-deterministic finite automaton. We proposed an iterative algorithm, which includes two rules for updating the status (or, more generally, the cost) of nodes in an appropriately constructed observer for the system, with inputs and outputs. We developed this algorithm to choose strategies that more efficiently lead us to a situation where the state of the system is known exactly.

The iterative algorithm is our proposal for a solution to this kind of problems. One should keep in mind, that once we achieve our goal and arrive at a singleton set of state estimates, a next step might be to remain in cycles of strictly singleton sets (i.e., for the system to continue operating while its state remains known).

After this procedure, we obtain the cost of each state and then we choose which path has the lowest cost to reach a singleton state. There is a possibility to have more than one strategy with the same cost.

5.2 Future Work

An interesting future extension of this problem is to apply it to the case of partially defined systems. The challenge in this case is that deadlocks might exist. Also it will be interesting to determine whether (approximate) solutions to the problem can be obtained with less complexity using a detector [10] as opposed to an observer. Such an approach may have reduced complexity at the cost of being suboptimal.

Another interesting extension is to formulate the problem as a stochastic system (i.e., with probabilities on inputs or outputs). An important question that arises is how the strategy changes when we know that one output is more likely to occur compared to another. In such case, one would also need to adopt appropriate (stochastic) criteria for optimality.

Bibliography

[1] C. N. Hadjicostis, *Estimation and Inference in Discrete Event Systems*. Springer, 2020.

[2] R. L. Rivest and R. E. Schapire, "Inference of finite automata using homing sequences," *Information and Computation*, vol. 103, no. 2, pp. 299–347, 1993.

[3] J. Dubreil, P. Darondeau, and H. Marchand, "Opacity enforcing control synthesis," in *Proceedings of 9th International Workshop on Discrete Event Systems*, pp. 28–35,2008.

[4] A. Saboori and C. N. Hadjicostis, "Opacity-enforcing supervisory strategies via state estimator constructions," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1155–1165, 2011.

[5] Y. Falcone and H. Marchand, "Runtime enforcement of K-step opacity," in *Proceedings of 52nd IEEE Conference on Decision and Control (CDC)*, pp. 7271–7278, 2013.

[6] J. Park and S. A. Reveliotis, "Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings," *IEEE Transactions on Automatic Control*, vol. 46, no. 10, pp. 1572–1583, 2001.

[7] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer Science & Business Media, 2009.

[8] S. Shu, F. Lin, and H. Ying, "Detectability of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 12, pp. 2356–2359, 2007.

[9] S. Shu and F. Lin, "Detectability of discrete event systems with dynamic event observation," *Systems & Control Letters*, vol. 59, no. 1, pp. 9–17, 2010.

[10] C. N. Hadjicostis and C. Seatzu, "K- detectability in discrete event systems," in *Proceedings of 55th IEEE Conference on Decision and Control (CDC)*, pp. 420–425, 2016.

[11] S. Shu and F. Lin, "Enforcing detectability in controlled discrete event systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 8, pp. 2125–2130, 2013.

[12] M. Pocci, I. Demongodin, N. Giambiasi, and A. Giua, "Synchronizing sequences on a class of unbounded systems using synchronized Petri nets," *Discrete Event Dynamic Systems*, vol. 26, no. 1, pp. 85–108, 2016.

[13] C. Wu, I. Demongodin, and A. Giua, "Correction to "Synchronizing sequences on a class of unbounded systems using synchronized Petri nets"," *Discrete Event Dynamic Systems*, vol. 29, no. 4, pp. 521–526, 2019.

[14] N. Yevtushenko, V. Kuliamin, and N. Kushik, "Evaluating the complexity of deriving adaptive homing, synchronizing and distinguishing sequences for nondeterministic FSMs," in *Proceedings of IFIP International Conference on Testing Software and Systems*, pp. 86–103, Springer, 2019.

[15] D. Lee and M. Yannakakis, "Testing finite-state machines: state identification and verification," *IEEE Transactions on Computers*, vol. 43, no. 3, pp. 306–320, 1994.

[16] N. Kushik, K. El-Fakih, N. Yevtushenko, and A. R. Cavalli, "On adaptive experiments for nondeterministic finite state machines," *International Journal on Software Tools for Technology Transfer*, vol. 18, no. 3, pp. 251–264, 2016.

[17] M. Soucha, "Finite state machine state identification sequences," *Open Informatics, Computer and Information Science, Faculty of Electrical Engineering, Department of Cybernetics*, 2014.

[18] X. Yin and S. Lafortune, "A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 8, pp. 2140–2154, 2016.