2

TAXONOMY AND THEORY OF BEHAVIORS

The goal of this chapter is to provide a unified notational framework for the book, and to introduce key elements of a theory for functional synthesis of FSMs. First we define some useful classes of finite state machines (FSMs) and finite automata (FAs), and investigate their inter-relationship. We will show that a non-deterministic FSM (NDFSM) can be used to specify a set of behaviors. Then we will describe how different behaviors can be explored within an NDFSM specification. These concepts are central to exact algorithms for state minimization for FSMs. At the end of the chapter, the correctness of our state minimization algorithms will be proved for a class of FSMs called pseudo non-deterministic FSMs.

2.1 TAXONOMY OF FINITE STATE MACHINES

We use characteristic functions proposed by Cerny [19] to represent sets and relations, both in theory and in practice. In the sequel, $B = \{0, 1\}$.

Definition 2.1 *Given a subset $S \subseteq U$ where U is some finite domain, the* **characteristic function of** *S, $\chi_S : U \rightarrow B$, is defined as follows. For each element $x \in U$,*

$$\chi_S(x) = \begin{cases} 0 \text{ if } x \notin S, \\ 1 \text{ if } x \in S. \end{cases}$$

T. Kam et al., *Synthesis of Finite State Machines*

Definition 2.2 *Given a relation $R \subseteq X \times Y$ where X and Y are some finite domains, the* **characteristic function of** R $\chi_R : X \times Y \rightarrow B$, *is defined as follows. For each pair $(x, y) \in X \times Y$,*

$$\chi_R(x,y) = \begin{cases} 0 \text{ if } x \text{ and } y \text{ are not in relation } R, \\ 1 \text{ if } x \text{ and } y \text{ are in relation } R. \end{cases}$$

The above definition can be extended to any n-ary relation.

Definition 2.3 *A* **non-deterministic FSM** *(NDFSM), or simply an* **FSM,** *is defined as a 5-tuple $M = \langle S, I, O, T, R \rangle$ where S represents the finite state space, I represents the finite input space and O represents the finite output space. T is the transition relation defined as a characteristic function $T : I \times S \times S \times O \rightarrow B$. On an input i, the NDFSM at present state p can transit to a next state n and output o if and only if $T(i,p,n,o) = 1$ (i.e., (i,p,n,o) is a transition). There exists one or more transitions for each combination of present state p and input i. $R \subseteq S$ represents the set of reset states.* [1]

Note that in this and subsequent definitions, the state space S, the input space I and the output space O can be generic discrete spaces and so S, I and O can assume symbolic values [37, 99]. A special case is when S, I and O are the Cartesian product of copies of the space $B = \{0, 1\}$, i.e., they are binary variables. The fact that the FSMs have symbolic versus binary encoded input and output variables does not change the formulation of problem. The theory extends in a straightforward manner to encoded state spaces.

The above is the most general definition of an FSM and it contains, as special cases, different well-known classes of FSMs. An FSM can be specified by a state transition table (STT) which is a tabular list of the transitions in T. An FSM defines a transition structure that can also be described by a state transition graph (STG). By a labeled edge $p \xrightarrow{i/o} n$, the FSM transits from state p on input i to state n with output o.

Definition 2.4 *Given an FSM $M = \langle S, I, O, T, R \rangle$, the* **state transition graph** *of M is a labeled directed graph, $STG(M) = \langle V, E \rangle$, where each state $s \in S$ corresponds to a vertex in V labeled s and each transition $(i,p,n,o) \in T$ corresponds to a directed edge in E from vertex p to vertex n, and the edge is labeled by the input/output pair i/o.*

[1] In subsequent definitions, R represents the set of reset states while r represents the unique reset state.

To capture flexibility/choice/don't-care/non-determinism in the next state n and/or the output o from a present state p on an input i, one can specify one or more transitions $(i, p, n, o) \in T$. As said above, we assume that the state transition relation T is complete with respect to i and p, i.e., there is always at least one transition out of each state on each input. This differs from the situation in formal verification where incomplete automata are considered.

Definition 2.5 *A state transition relation T is* **complete** *if*

$$\forall i, p \; \exists n, o \; [T(i, p, n, o)] = 1.$$

$\forall$ is used to denote universal quantification, whereas $\exists$ denotes existential quantification. The above equation is an abbreviation of the following statement:

$$\forall i \in I \; \forall p \in S \; \exists n \in S \; \exists o \in O \text{ such that } T(i, p, n, o) = 1$$

The relational representation of T allows non-deterministic transitions with respect to next states and/or outputs, and also allows correlations between next states and outputs. More specialized forms of FSMs are derived by restricting the form of transitions allowed in T. FSMs can be categorized by answering the following questions:

1. Instead of a single transition relation, can the next state information and output information of the FSM be represented separately by two functions or relations?
2. Can the FSM be represented by functions instead of relations? [2]
3. Are the next states and outputs in the relations and/or in the functions correlated? Next state information is not correlated with the output if the former is related to the inputs and present states only. Uncorrelated output is similarly defined.

Now we introduce different classes of FSMs. An NDFSM is a pseudo non-deterministic FSM (PNDFSM) if for each triple $(i, p, o) \in I \times S \times O$, there is a unique state n such that $T(i, p, n, o) = 1$. It is non-deterministic because for a given input and present state, there may be more than one possible output;

[2] Relations are denoted by upper-case letters (e.g., Δ denotes a next state relation) while functions are denoted by lower-case ones (e.g., δ represents a next state function).

it is called pseudo non-deterministic because edges (i.e., transitions) carrying different outputs must go to different next states [3].

Definition 2.6 *A* **pseudo non-deterministic FSM** *(PNDFSM) is a 6-tuple* $M = \langle S, I, O, \delta, \Lambda, R \rangle$. *$S$ represents the finite state space, I represents the finite input space and O represents the finite output space. δ is the next state function defined as $\delta : I \times S \times O \rightarrow S$ where each combination of input, present state and output is mapped to a unique next state. Λ is the output relation defined as a characteristic function $\Lambda : I \times S \times O \rightarrow B$ where each combination of input and present state is related to one or more outputs. $R \subseteq S$ represents the set of reset states.*

In other words for a PNDFSM, $T(i,p,n,o) = 1$ if and only if $\Lambda(i,s,o) = 1$ and $n = \delta(i,s,o)$. Since the next state n is unique for a given combination of input, present state and output, it can be given by a next state function $n = \delta(i,p,o)$. Since the output is non-deterministic in general, it is represented by the relation Λ.

Theorem 2.1 *An NDFSM is* **pseudo non-deterministic** *if*

$$\forall i,p,o \; !n \; [T(i,p,n,o)] = 1$$

where ! denotes a new operator called the *unique* quantifier which will be introduced in Section 3.3.2. $!x \; F(x,y) \stackrel{\text{def}}{=} \{y | y \text{ is related to a unique } x \text{ in } F\}$.

One can extend the previous definition to get a numerable family of machines as follows. An NDFSM is a k-step pseudo non-deterministic FSM (k-PNDFSM), where $k \in \omega$ (i.e., k is a natural number), if at any present state, the choice of the next state can be uniquely identified by observing input-output sequences of length up to k. An NDFSM is a k-step pseudo non-deterministic FSM (k-PNDFSM), where $k \in \omega$, if for each tuple $(i, i_2, \ldots, i_k, p, o, o_2, \ldots, o_k) \in I \times I \times \cdots \times I \times S \times O \times O \times \cdots \times O$, there is a unique next state $n \in S$ and there are states $s_2, \cdots, s_k \in S$ such that $T(i,p,n,o) = 1$ and $T(i_2, n, s_2, o_2) = T(i_3, s_2, s_3, o_3) = \cdots = T(i_k, s_{k-1}, s_k, o_k) = 1$.

Definition 2.7 *A* **k-step pseudo non-deterministic FSM** *(k-PNDFSM) is a 6-tuple* $M = \langle S, I, O, \delta, \Lambda, R \rangle$. *$\delta$ is the next state function defined as δ :*

[3] The underlying finite automaton of a PNDFSM is deterministic.

$I \times \cdots \times I \times S \times O \times \cdots \times O \rightarrow S$ *where each combination of present state, k inputs and k outputs gives a unique next state.* Λ *is the output relation defined as a characteristic function* $\Lambda : I \times S \times O \rightarrow B$ *where each combination of input and present state is related to one or more outputs.* $R \subseteq S$ *represents the set of reset states.*

k-PNDFSMs are sufficient in many applications to express all existing non-determinism. By definition, a PNDFSM is an 1-PNDFSM. Cerny in [21] has given a polynomial algorithm to convert a k-PNDFSM to a PNDFSM. A k-PNDFSM has a representation smaller or equal to that of an equivalent PNDFSM. We shall consider the state minimization of a 1-PNDFSM only and assume that the k-PNDFSM can be handled by first going through such a conversion step.

Classical texts usually describe the Mealy and Moore model of FSMs. For completeness, they are also defined here as subclasses of NDFSM. A Mealy NDFSM is an NDFSM where there exists a next state relation [4] $\Delta : I \times S \times S \rightarrow B$ and an output relation [5] $\Lambda : I \times S \times O \rightarrow B$ such that for every $(i, p, n, o) \in I \times S \times S \times O$, $T(i, p, n, o) = 1$ if and only if $\Delta(i, p, n) = 1$ and $\Lambda(i, p, o) = 1$.

Definition 2.8 *A* **Mealy NDFSM** *is a 6-tuple* $M = \langle S, I, O, \Delta, \Lambda, R \rangle$. S *represents the finite state space,* I *represents the finite input space and* O *represents the finite output space.* Δ *is the next state relation defined as a characteristic function* $\Delta : I \times S \times S \rightarrow B$ *where each combination of input and present state is related to a non-empty set of next states.* Λ *is the output relation defined as a characteristic function* $\Lambda : I \times S \times O \rightarrow B$ *where each combination of input and present state is related to a non-empty set of outputs.* $R \subseteq S$ *represents the set of reset states.*

Note that next states and outputs are not correlated in the state transition relation of a Mealy machine.

A Moore NDFSM is an NDFSM where there exists a next state relation $\Delta : I \times S \times S \rightarrow B$ and an output relation $\Lambda : S \times O \rightarrow B$ such that for all

[4] The next state relation Δ can be viewed as a function $\Delta : I \times S \rightarrow 2^S$; $n \in \Delta(i, p)$ if and only if n is a possible next state of state p on input i.

[5] The output relation Λ can be viewed as a function $\Lambda : I \times S \rightarrow 2^O$; $o \in \Lambda(i, p)$ if and only if o is a possible output of state p on input i.

$(i,p,n,o) \in I \times S \times S \times O$, $T(i,p,n,o) = 1$ if and only if $\Delta(i,p,n) = 1$ and $\Lambda(p,o) = 1$.

Definition 2.9 *A* **Moore NDFSM** *is a 6-tuple* $M = \langle S, I, O, \Delta, \Lambda, R \rangle$. *$S$ represents the finite state space, I represents the finite input space and O represents the finite output space. Δ is the next state relation defined as a characteristic function $\Delta : I \times S \times S \rightarrow B$ where each combination of input and present state is related to a non-empty set of next states. Λ is the output relation defined as a characteristic function $\Lambda : S \times O \rightarrow B$ where each present state is related to a non-empty set of outputs. $R \subseteq S$ represents the set of reset states.*

As a special case of Mealy machine, outputs of Moore machines depend only on the present state (but not on inputs).

The definition of Moore machine presented here is the same as the one given by Moore in [84] and followed by authors in the field (e.g., [118]). The key fact to notice is that the output is associated with the present state. In other words, the common output associated with a given state, goes on all transitions that leave that state. This is a reasonable assumption when modeling a hardware system. However, it is common to find in textbooks [60, 49] a "dual" definition where the output is associated with the next state. In other words, the common output associated with a given state, is on all edges that go into that state, while edges leaving a given state may carry different outputs.

This second definition enjoys the nice property that it is always possible to convert a Mealy machine into a Moore machine. This may be the reason for its choice in computer science textbooks. Instead with the first definition, there are Mealy machines that have no Moore equivalent. For example, a wire can be consider a Mealy machine with one state and with its input connected directly to its output. It does not have an equivalent Moore machine.

An NDFSM is an incompletely specified FSM (ISFSM) if for each pair $(i,p) \in I \times S$ such that $T(i,p,n,o) = 1$, (1) the machine can transit to a unique next state n or to any next state, and (2) the machine can produce a unique output o or produce any output. But an ISFSM is not allowed to transit to a strict subset of the states, nor produce a strict subset of the outputs.

Definition 2.10 *An* **incompletely specified FSM** *(ISFSM) can be defined as a 6-tuple* $M = \langle S, I, O, \Delta, \Lambda, R \rangle$. *$S$ represents the finite state space, I rep-*

resents the finite input space and O represents the finite output space. Δ is the next state relation defined as a characteristic function $\Delta : I \times S \times S \rightarrow B$ where each combination of input and present state is related to a single next state or to all states. Λ is the output relation defined as a characteristic function $\Lambda : I \times S \times O \rightarrow B$ where each combination of input and present state is related to a single output or to all outputs. $R \subseteq S$ represents the set of reset states.

Incomplete specification is used here to express some types of don't cares in the next states and/or outputs. We warn that even though "incompletely specified" is established terminology in the sequential synthesis literature, it conflicts with the fact that ISFSMs have a transition relation T that is actually completely specified with respect to present state p and input i, because there is at least one transition for each (i, p) pair in T.

A deterministic FSM (DFSM) or completely specified FSM (CSFSM) is an NDFSM where for each pair $(i, p) \in I \times S$, there is a unique next state n and a unique output o such that $T(i, p, n, o) = 1$, i.e., there is a unique transition from (i, p). In addition, R contains a unique reset state.

Definition 2.11 *A* **deterministic FSM** *(DFSM) or* **completely specified FSM** *(CSFSM) can be defined as a 6-tuple $M = \langle S, I, O, \delta, \lambda, r \rangle$. S represents the finite state space, I represents the finite input space and O represents the finite output space. δ is the next state function defined as $\delta : I \times S \rightarrow S$ where $n \in S$ is the next state of present state $p \in S$ on input $i \in I$ if and only if $n = \delta(i, p)$. λ is the output function defined as $\lambda : I \times S \rightarrow O$ where $o \in O$ is the output of present state $p \in S$ on input $i \in I$ if and only if $o = \lambda(i, p)$. $r \in S$ represents the unique reset state.*

Note that an ISFSM and a DFSM are both next-state output uncorrelated because we can represent the next state and output information separately. But a PNDFSM (and k-PNDFSM) is next-state output correlated as the next state is correlated with the output by $n = \delta(i, p, o)$.

A Moore DFSM is a Moore NDFSM where for each pair $(i, p) \in I \times S$, there is a unique next state n and for each $p \in S$, a unique output o such that $T(i, p, n, o) = 1$. In addition, R contains a unique reset state.

Definition 2.12 *A* **Moore DFSM** *can be defined as a 6-tuple $M = \langle S, I, O, \delta, \lambda, r \rangle$. S represents the finite state space, I represents the finite input*

space and O represents the finite output space. δ is the next state function defined as $\delta : I \times S \rightarrow S$ where $n \in S$ is the next state of present state $p \in S$ on input $i \in I$ if and only if $n = \delta(i, p)$. λ is the output function defined as $\lambda : S \rightarrow O$ where $o \in O$ is the output of present state $p \in S$ if and only if $o = \lambda(p)$. $r \in S$ represents the reset state.

Definition 2.13 *Given an NDFSM, a state s is* **output-deterministic** *if for every input i, there exists a unique output o such that s outputs o under input i. An NDFSM is* **output-deterministic** *(i.e., the NDFSM is an ODFSM) if every reachable state is output-deterministic.*

Theorem 2.2 *An NDFSM is output-deterministic if and only if*

$$\forall i, p \; !o \; \exists n \; [T(i, p, n, o)] = 1.$$

Definition 2.14 *An NDFSM is* **spurious non-deterministic** *if for each input sequence, there is a unique output sequence.*

A spurious non-deterministic FSM, as opposed to a true non-deterministic one, can be viewed as a DFSM. Is spurious non-determinism the same as output-determinism for NDFSMs?

Theorem 2.3 *If an NDFSM is spurious non-deterministic, then all its reachable states are output-deterministic.*

Proof: Given an NDFSM that is spurious non-deterministic. Assume that a reachable state s is not output-deterministic, there is an input i on which s can produce two different outputs. As s is reachable, say with input sequence σ_i from a reset state, the machine will produce different output sequences under the input sequence $\sigma_i i$. As this contradicts the fact that the NDFSM is spurious non-deterministic, our assumption was false. ■

2.2 TAXONOMY OF FINITE AUTOMATA

Definition 2.15 *Given a finite set of symbols Σ (i.e., a finite alphabet), a* **language** *$\mathcal{L}$ is a set of strings of symbols from Σ.*

Definition 2.16 *A* **deterministic finite automaton** *(DFA), or simply* **finite automaton** *(FA), is defined as a 5-tuple* $A = \langle S, \Sigma, \delta, r, F \rangle$. *$S$ represents the finite state space and Σ represents a finite alphabet. δ is the next state function defined as* $\delta : \Sigma \times S \rightarrow S$ *where* $n \in S$ *is the next state of present state* $p \in S$ *on symbol* $i \in \Sigma$ *if and only if* $n = \delta(i, p)$. $r \in S$ *represents the reset state.* $F \subseteq S$ *is the set of final states.*

The next state function δ can be extended to have as argument strings in Σ^*, (i.e., $\delta : \Sigma^* \times S \rightarrow S$) by $\delta(\rho i, s) = \delta(i, \delta(\rho, s))$.

A string x is said to be **accepted** *by the DFA A if* $\delta(x, r)$ *is a state in F. The* **language** *accepted by A, designated* $\mathcal{L}(A)$, *is the set of strings* $\{x | \delta(x, r) \in F\}$.

Definition 2.17 *A* **non-deterministic finite automaton** *(NDFA) is defined as a 5-tuple* $A = \langle S, \Sigma, \Delta, r, F \rangle$. *$S$ represents the finite state space and Σ represents the finite alphabet. Δ is the next state relation defined as a characteristic function* $\Delta : \Sigma \times S \times S \rightarrow B$ *where* $n \in S$ *is a next state of present state* $p \in S$ *on symbol* $i \in \Sigma$ *if and only if* $n \in \Delta(i, p)$ [6]. $r \in S$ *represents the reset state.*

The next state relation can be extended to have as argument strings in Σ^* (i.e., $\Delta : \Sigma^* \times S \times S \rightarrow B$) as follows: $\Delta(\rho i, s, s'') = 1$ *if and only if there exists* $s' \in S$ *such that* $\Delta(\rho, s, s') = 1$ *and* $\Delta(i, s', s'') = 1$.

A string x is said to be **accepted** *by the NDFA A if there exists a sequence of transitions corresponding to x such that* $\Delta(x, r)$ *contains a state in F. The* **language** *accepted by A, designated* $\mathcal{L}(A)$, *is the set of strings* $\{x | \Delta(x, r) \cap F \neq \emptyset\}$.

Theorem 2.4 *Let $\mathcal{L}$ be the language accepted by a non-deterministic finite automaton. Then there exists a deterministic finite automaton that accepts $\mathcal{L}$.*

Proof: By subset construction [49]. ■

Corollary 2.5 *DFAs and NDFAs (and NDFAs with ϵ-transitions and regular expressions) accept the same class of languages.*

[6] The next state relation Δ can be viewed as a function $\Delta : I \times S \rightarrow 2^S$; $n \in \Delta(i, p)$ if and only if n is a possible next state of state p on input i.

Proof: The class of languages accepted by NDFAs includes the set of languages accepted by DFAs (regular sets). Converse is also true because of the previous theorem. ■

As DFAs and NDFAs accept the same set of languages, we shall refer to both as finite automata (FAs). However given an NDFA, its equivalent DFA may have an exponential number of states as compared with the NDFA.

Whereas a DFA is restricted to have exactly one transition from a state for each symbol, an NDFA may have any number of transitions for a given symbol, including zero transition. If when processing a string, an NDFA comes to a state from which there is no transition labeled with a symbol, the path followed is terminated. Actually, given an NDFA with missing transitions, an equivalent NDFA without any missing transition can be constructed by adding a non-accepting (i.e., not final) dummy state, and pointing all missing transitions to this dummy state.

An NDFA allows multiple reset states. Again given an NDFA with multiple reset states, an equivalent NDFA can be constructed with a new unique reset state.

2.3 CONVERSIONS BETWEEN FINITE STATE MACHINES AND FINITE AUTOMATA

Theorem 2.6 (DFA → DFSM) *Given a DFA* $A = \langle S, I, \delta, r, F \rangle$*, a DFSM* $M = \langle S, I, O, \delta, \lambda, r \rangle$ *can be constructed such that* $O = \{0, 1\}$ *and for every* $i \in I$ *and* $s \in S$*,* $\lambda(i, s) = 1$ *if and only if* $\delta(i, s) \in F$*.*

The DFSM M will output 1 after an input string if and only if the string read since the reset state is also accepted by the DFA A. We ignore the ϵ*-string (i.e., the fact that the reset state of A may be accepting, i.e., final) as it cannot be modeled by a Mealy DFSM.*

Theorem 2.7 (NDFSM → NDFA) *Given an NDFSM* $M = \langle S, I, O, T, R \rangle$*, an NDFA* $A = \langle S, I \times O, \Delta, R, S \rangle$ *can be constructed such that for each* $io \in I \times O$ *(i.e., a symbol containing an input and an output part) and for each* $s \in S$*,*

$\Delta(io, p, n) = 1$ *if and only if* $T(i, p, n, o) = 1$. *All states are accepting (i.e., final) in A.*

Theorem 2.8 *An FSM is pseudo non-deterministic if and only if the FA associated with it is deterministic.*

Proof: Note that when in the FSM from a state there is no edge with a given input/output label, in the constructed finite automaton one adds a transition with that label to a dummy non-accepting state. Non-determinism arises only when in the FSM from a state there is more than one edge with the same input/output label. By definition this cannot happen in a PNDFSM, while it can happen in a more general NDFSM. ■

Theorem 2.9 (NDFSM → PNDFSM) *Given an NDFSM* $M = \langle S, I, O, T, R \rangle$, *an NDFA* $A = \langle S, I \times O, \Delta, R, S \rangle$ *can be constructed according to the above Theorem 2.7. The NDFA can then be "determinized" via subset construction [49] to obtain a DFA* $A^{det} = \langle 2^S, I \times O, \Delta^{det}, R^{det}, 2^S \rangle$. *Then consider an NDFSM* $M^{det} = \langle 2^S, I, O, T^{det}, R^{det}, 2^S \rangle$ *where each symbol of* A^{det} *is split into an input and an output, and* $T^{det}(i, p, n, o) = 1$ *if and only if* $\Delta^{det}(io, p, n) = 1$. M^{det} *is pseudo non-deterministic.*

Theorem 2.10 (k-PNDFSM → 1-PNDFSM) *Construction given by Cerny in [21].*

2.4 TRACE SETS AND BEHAVIORS

In the remaining sections of this chapter, we shall introduce different concepts that lead to an exact algorithm for state minimization. Our discussion will be based on state minimization of PNDFSMs because we believe it is the largest subclass of NDFSMs that our state minimization procedure can handle correctly. Miller has given a proof in [82] for the classical exact algorithm for ISFSM state minimization [43]. We contribute a concise proof for our exact state minimization algorithm for PNDFSMs, by presenting a number of theorems in the following sections. A different rigorous proof can be found also in [119]. In this section, we first show that a DFSM realizes a behavior while an NDFSM realizes a set of behaviors.

Definition 2.18 *Given a finite set of inputs I and a finite set of outputs O, a* **trace** *between I and O is a pair of input and output sequences (σ_i, σ_o) where $\sigma_i \in I^*$, $\sigma_o \in O^*$ and $|\sigma_i| = |\sigma_o|$.*

Definition 2.19 *A* **trace set** *is simply a set of traces.*

Definition 2.20 *An NDFSM $M = \langle S, I, O, T, R \rangle$* **realizes** *a trace set between I and O from state $s_0 \in S$, denoted by $\mathcal{L}(M|_{s_0})$ [7], if for every trace $(\{i_0, i_1, \ldots, i_j\}, \{o_0, o_1, \ldots, o_j\})$ in the trace set, there exists a state sequence $s_1, s_2, \ldots, s_{j+1}$ such that $\forall k : 0 \leq k \leq j$, $T(i_k, s_k, s_{k+1}, o_k) = 1$.*

Definition 2.21 *An ISFSM $M = \langle S, I, O, \Delta, \Lambda, R \rangle$* **realizes** *a trace set between I and O from state $s_0 \in S$, denoted by $\mathcal{L}(M|_{s_0})$, if for every trace $(\{i_0, i_1, \ldots, i_j\}, \{o_0, o_1, \ldots, o_j\})$ in the trace set, there exists a state sequence $s_1, s_2, \ldots, s_{j+1}$ such that $\forall k : 0 \leq k \leq j$,*

- $s_{k+1} \in \Delta(i_k, s_k)$, *and*
- $o_k \in \Lambda(i_k, s_k)$.

The trace set realized by a deterministic FSM with inputs I and outputs O is called a behavior between the inputs I and the outputs O. A formal definition follows.

Definition 2.22 *Given a finite set of inputs I and a finite set of outputs O, a* **behavior** *between I and O is a trace set, $\mathcal{B} = \{(\sigma_i, \sigma_o) \mid |\sigma_i| = |\sigma_o|\}$, which satisfies the following conditions:*

1. *Completeness:*
   For an arbitrary sequence σ_i on I, there exists a unique pair in $\mathcal{B}$ whose input sequence is equal to σ_i.
2. *Regularity:*
   There exists a DFSM $M = \langle S, I, O, \delta, \lambda, s_0 \rangle$ such that, for each $((i_0, \ldots, i_j), (o_1, \ldots, o_j)) \in \mathcal{B}$, there is a sequence of states $s_1, s_2, \ldots, s_{j+1}$ with the property that $s_{k+1} = \delta(i_k, s_k)$ and $o_k = \lambda(i_k, s_k)$ for every $k : 0 \leq k \leq j$.

[7] If the NDFSM M is viewed as a NFA A which alphabet is $\Sigma = I \times O$, the trace set of M from a state s_0 corresponds to the language of A from s_0, and both will be denoted by $\mathcal{L}(M|_{s_0})$.

For each state in a deterministic FSM, each input sequence corresponds to exactly one possible output sequence. Given a reset state, a deterministic FSM realizes a unique input-output behavior. But given a behavior, there can be (possibly infinitely) many DFSMs that realize the behavior. Thus, the mapping between behaviors and DFSM realizations is a one-to-many relation.

Any other kind of FSMs, on the other hand, can represent a set of behaviors because by different choices of next states and/or outputs, more than one output sequence can be associated with an input sequence. Moreover, multiple reset states allow alternative trace sets to be specified; depending on the choice of the reset state, a behavior within the trace set from the chosen reset state can be implemented. Therefore, while a DFSM represents a single behavior, a nondeterministic FSM (NDFSM) can be viewed as representing a set of behaviors. Each such behavior within its trace set is called a contained behavior of the NDFSM. Then an NDFSM expresses handily flexibilities in sequential synthesis. Using an NDFSM, a user can specify that one out of the set of behaviors is to be implemented. The choice of a particular behavior for implementation is based on some cost function such as the number of states. Other cost functions related to implementability will be investigated in Chapter 8.

2.5 MACHINE CONTAINMENT

The fact that an NDFSM represents a set of behaviors leads naturally to the notion of behavioral containment or trace set inclusion between NDFSMs.

Definition 2.23 *An NDFSM* $M = \langle S, I, O, T, R \rangle$ **behaviorally contains** *another NDFSM* $M' = \langle S', I, O, T', R' \rangle$, *denoted by* $\mathcal{L}(M) \supseteq \mathcal{L}(M')$, *if*[8] *for every* $r' \in R'$, *there exists* $r \in R$ *such that the trace set of* M *from* r *contains the trace set of* M' *from* r'. *i.e.,*

$$\mathcal{L}(M) \supseteq \mathcal{L}(M') \text{ if and only if } \forall r' \in R' \; \exists r \in R \; \mathcal{L}(M|_r) \supseteq \mathcal{L}(M'|_{r'}).$$

A different and more restrictive notion of containment is STG-containment which requires that the STG of the contained machine M' is a subgraph of the STG of the containing machine M and implies that $S \supseteq S'$ [9].

[8] *cf.* classical definition for ISFSM minimization.

[9] Notice that this definition is different from the notion of structural containment, as defined by various authors (for instance, in [119], pag.90).

Definition 2.24 *An NDFSM* $M = \langle S, I, O, T, R \rangle$ **STG-contains** *another NDFSM* $M' = \langle S', I, O, T', R' \rangle$, *denoted by* $STG(M) \supseteq STG(M')$, *if there exists a one-to-one mapping* $\varphi : S' \rightarrow S$ *such that*

1. $\forall r \in R'\ \varphi(r) \in R$, *and*
2. $\forall i \in I\ \forall p \in S'\ \forall q \in S'\ \forall o \in O$ *if* $T'(i, p, q, o) = 1$ *then* $T(i, \varphi(p), \varphi(q), o) = 1$.

Theorem 2.11 *If* $STG(M) \supseteq STG(M')$ *then* $\mathcal{L}(M) \supseteq \mathcal{L}(M')$. *The converse is not true in general.*

The implication is obvious because $STG(M) \supseteq STG(M')$ means that M' simulates exactly a fragment of M, so the language of M' must be a subset of the language of M. The converse is false already for ISFSMs, because for instance state splitting is not captured by STG-containment. This theorem implies that, in general, we cannot explore all behaviors contained in the trace set of an NDFSM by enumerating DFSMs which are STG-contained within the NDFSM. But in Section 2.8, we shall show that all behaviors contained by a PNDFSM can be explored by finding DFSMs that are STG-contained in a derived machine called the power NDFSM.

2.6 BEHAVIORAL EXPLORATION IN PNDFSMS

In this section, we introduce the notion of a closed cover and establish the fact that by finding all closed covers of a PNDFSM, we can explore all behaviors contained in it.

Definition 2.25 *Given an NDFSM* $M = \langle S, I, O, T, R \rangle$, *a set of state sets,* $\{c_1, c_2, \ldots, c_n\}$, *is a* **cover** *of* M *if*[10] *there exists* $r \in R$ *and* $c_j : 1 \leq j \leq n$ *such that* $r \in c_j$.

Definition 2.26 *Given an NDFSM* $M = \langle S, I, O, T, R \rangle$, *a set of state sets,* $K = \{c_1, c_2, \ldots, c_n\}$, *is* **closed** *in* M *if for every* $i \in I$ *and* $c_j : 1 \leq j \leq n$,

[10] *cf.* classical definition for ISFSM minimization.

there exists $o \in O$ *and* $c_k : 1 \leq k \leq n$ *such that for each* $s \in c_j$, *there exists* $s' \in c_k$ *such that* $T(i,s,s',o) = 1$. *i.e.,*

$$\forall i \in I\ \forall c_j \in K\ \exists o \in O\ \exists c_k \in K\ \forall s \in c_j\ \exists s' \in c_k\ T(i,s,s',o) = 1.$$

The above definition applies to all kinds of FSMs. The following lemmas derive a characterization equivalent to closure condition for the restricted cases when M is, respectively, a PNDFSM and an ISFSM.

Lemma 2.12 *Given a PNDFSM* $M = \langle S, I, O, \delta, \Lambda, R\rangle$, *a set of state sets,* $K = \{c_1, c_2, \ldots, c_n\}$, *is* **closed** *in* M *if and only if for every* $i \in I$ *and* $c_j : 1 \leq j \leq n$, *there exists* $o \in O$ *such that*

1. *for each* $s \in c_j$, $\Lambda(i,s,o) = 1$, *and*
2. *there exists* $c_k : 1 \leq k \leq n$ *such that for every* $s \in c_j$, $\delta(i,s,o) \in c_k$.

i.e., $\forall i \in I\ \forall c_j \in K\ \exists o \in O\ \{\ [\forall s \in c_j\ \Lambda(i,s,o) = 1] \cdot [\exists c_k \in K\ \forall s \in c_j\ \delta(i,s,o) \in c_k]\ \}$.

Proof: By the definition of PNDFSM, $[T(i,s,s',o) = 1] \Leftrightarrow [\Lambda(i,s,o) = 1] \cdot [s' = \delta(i,s,o)]$. The closure condition for PNDFSM can be derived by a series of quantifier moves starting from the formula in Definition 2.26.

$$\begin{aligned}
& \forall i \in I\ \forall c_j \in K\ \exists o \in O\ \exists c_k \in K\ \forall s \in c_j\ \exists s' \in c_k\ T(i,s,s',o) = 1 \\
\Leftrightarrow\ & \forall i \in I\ \forall c_j \in K\ \exists o \in O\ \exists c_k \in K\ \forall s \in c_j\ \exists s' \in c_k\ \{\ [\Lambda(i,s,o) = 1] \cdot [s' = \delta(i,s,o)]\ \} \\
\Leftrightarrow\ & \forall i \in I\ \forall c_j \in K\ \exists o \in O\ \exists c_k \in K\ \forall s \in c_j\ \{\ [\Lambda(i,s,o) = 1] \cdot [\exists s' \in c_k\ s' = \delta(i,s,o)]\ \} \\
\Leftrightarrow\ & \forall i \in I\ \forall c_j \in K\ \exists o \in O\ \exists c_k \in K\ \{\ [\forall s \in c_j\ \Lambda(i,s,o) = 1] \cdot [\forall s \in c_j\ \delta(i,s,o) \in c_k]\ \} \\
\Leftrightarrow\ & \forall i \in I\ \forall c_j \in K\ \exists o \in O\ \{\ [\forall s \in c_j\ \Lambda(i,s,o) = 1] \cdot [\exists c_k \in K\ \forall s \in c_j\ \delta(i,s,o) \in c_k]\ \}
\end{aligned}$$

■

Lemma 2.13 *Given an ISFSM* $M = \langle S, I, O, \Delta, \Lambda, R\rangle$, *a set of state sets,* $K = \{c_1, c_2, \ldots, c_n\}$, *is* **closed** *in* M *if and only if for every* $i \in I$ *and* $c_j : 1 \leq j \leq n$,

1. *there exists* $o \in O$ *such that for each* $s \in c_j$, $\Lambda(i,s,o) = 1$, *and*

2. *there exists* $c_k : 1 \leq k \leq n$ *such that for every* $s \in c_j$, *there exists* $s' \in c_k$ *such that* $\Delta(i, s, s') = 1$.

i.e., $\forall i \in I \; \forall c_j \in K \; \{ \; [\exists o \in O \; \forall s \in c_j \; \Lambda(i, s, o) = 1] \cdot [\exists c_k \in K \; \forall s \in c_j \; \exists s' \in c_k \; \Delta(i, s, s') = 1] \; \}$.

Proof: By the definition of ISFSM, $[T(i, s, s', o) = 1] \Leftrightarrow [\Lambda(i, s, o) = 1] \cdot [\Delta(i, s, s') = 1]$. The closure condition for ISFSM can be derived by a series of quantifier moves starting from the formula in Definition 2.26.

$$
\begin{aligned}
& \forall i \in I \; \forall c_j \in K \; \exists o \in O \; \exists c_k \in K \; \forall s \in c_j \; \exists s' \in c_k \; T(i, s, s', o) = 1 \\
\Leftrightarrow \; & \forall i \in I \; \forall c_j \in K \; \exists o \in O \; \exists c_k \in K \; \forall s \in c_j \; \exists s' \in c_k \; \{ \; [\Lambda(i, s, o) = 1] \cdot [\Delta(i, s, s') = 1] \; \} \\
\Leftrightarrow \; & \forall i \in I \; \forall c_j \in K \; \exists o \in O \; \exists c_k \in K \; \forall s \in c_j \; \{ \; [\Lambda(i, s, o) = 1] \cdot [\exists s' \in c_k \; \Delta(i, s, s') = 1] \; \} \\
\Leftrightarrow \; & \forall i \in I \; \forall c_j \in K \; \{ \; [\exists o \in O \; \forall s \in c_j \; \Lambda(i, s, o) = 1] \\
& \cdot [\exists c_k \in K \; \forall s \in c_j \; \exists s' \in c_k \; \Delta(i, s, s') = 1] \; \}
\end{aligned}
$$

■

Definition 2.27 *A set* K *of state sets is called a* **closed cover** *for* $M = \langle S, I, O, T, R \rangle$ *if*

1. K *is a cover of* M , *and*
2. K *is closed in* M.

Definition 2.28 *Let* $M = \langle S, I, O, T, R \rangle$, *and* $K = \{c_1, c_2, \ldots, c_n\}$ *be a closed cover for* M *where* $c_j \in 2^S$ *for* $1 \leq j \leq n$, *and* $M' = \langle S', I, O, T', R' \rangle$ *where* $S' = \{s_1, s_2, \ldots, s_n\}$.

K **is represented by** M' *(M' represents K) if for every* $i \in I$ *and* $j : 1 \leq j \leq n$, *there exists* $k : 1 \leq k \leq n$ *and* $o \in O$ *such that, if* $T'(i, s_j, s_k, o) = 1$ *then* $\forall s \in c_j \; \exists s' \in c_k \; T(i, s, s', o) = 1$.

Note that this definition implies a one-to-one mapping of K onto S'; in particular, $c_j \rightarrow s_j$ for $1 \leq j \leq n$. However, many different FSMs can represent a single closed cover.

Theorem 2.14 *If K is a closed cover for a PNDFSM M, then there exists a DFSM M' representing K. If K is represented by M', then $\mathcal{L}(M') \subseteq \mathcal{L}(M)$.*

Proof: Suppose $K = \{c_1, c_2, \ldots, c_n\}$ is a closed cover for $M = \langle S, I, O, T, R \rangle$. Then a DFSM $M' = \langle S', I, O, T', r' \rangle$ representing K can be constructed as follows. Let $S' = \{s_1, s_2, \ldots, s_n\}$ where there is a one-to-one correspondence between c_j and s_j for $1 \leq j \leq n$. As K is a cover, there exists a set $c_j : 1 \leq j \leq n$ and a reset state $r \in R$ such that $r \in c_j$. Pick one such c_j and choose r' to be the corresponding state s_j. As K is closed, for any $c_j : 1 \leq j \leq n$ under each $i \in I$, there exists $c_k : 1 \leq k \leq n$ and $o \in O$ such that $\forall s \in c_j \; \exists s' \in c_k \; T(i, s, s', o) = 1$. Out of these possible choices of c_k and o, we choose an arbitrary transition (i, s_j, s_k, o) for T' from present state s_j on input i. As each (i, s_j) pair has a unique transition, M' is a DFSM. With this definition of M', it follows directly that M' satisfies Definition 2.28 so that K is represented by M'.

For the second part of the theorem, suppose $K = \{c_1, c_2, \ldots, c_n\}$ is a closed cover for M, and K is represented by $M' = \langle S', I, O, T', r' \rangle$; then $S' = \{s_1, s_2, \ldots, s_n\}$ by Definition 2.28. To prove that $\mathcal{L}(M') \subseteq \mathcal{L}(M)$ we must show that for the unique reset state r' of M', $\exists r \in R$ such that $\mathcal{L}(M|_r) \supseteq \mathcal{L}(M'|_{r'})$. As K is a cover, $\exists r \in R$ such that $r \in c_j$ for some $c_j : 1 \leq j \leq n$. We show $\mathcal{L}(M|_r) \supseteq \mathcal{L}(M'|_{s_j})$ by induction on the length k of an arbitrary input sequence, $i_1, i_2, \ldots, i_k$. As M' is a DFSM, the input sequence produces from state s_j a unique output sequence $o_1, o_2, \ldots, o_k$ and a unique state sequence $s_{j(1)}, s_{j(2)}, \ldots, s_{j(k)}$. We claim that the same input sequence when applied to M at r will produce the same output sequence, via some state sequence $r_1, r_2, \ldots, r_k$, where $r_m \in c_{j(m)}$ for $1 \leq m \leq k$. Remember that $r \in c_j$. For the base case $k = 1$, as K is closed, there exists $c_{j(1)} : 1 \leq j(1) \leq n$ such that for every r_1 such that $T(i_1, r, r_1, o_1) = 1$, $r_1 \in c_{j(1)}$. As M is a PNDFSM, there is a unique $r_1 \in c_{j(1)}$ corresponding to output o_1. Assume the induction hypothesis is true for $k - 1$ where $k > 1$. So $r_{k-1} \in c_{j(k-1)}$. Again as K is closed, there exists $c_{j(k)} : 1 \leq j(k) \leq n$ such that for every r_k such that $T(i_k, r_{k-1}, r_k, o_k) = 1$, $r_k \in c_{j(k)}$ and r_k is unique. Thus the thesis is proved. ■

Theorem 2.15 *Given a PNDFSM M and a DFSM M', if $\mathcal{L}(M') \subseteq \mathcal{L}(M)$ then there exists a closed cover for M which is represented by M'.*

Proof: Let $M = \langle S, I, O, T, R \rangle$ and $M' = \langle S', I, O, T', R' \rangle$ where $S' = \{s_1, s_2, \ldots, s_n\}$. We assume that $\mathcal{L}(M') \subseteq \mathcal{L}(M)$. Let $K = \{c_1, c_2, \ldots, c_n\}$

be the set of state sets defined as follows. For every state $s_j \in S'$ in M', we construct the set of states c_j such that $s \in c_j$ if and only if the trace set of M from s contains the trace set of M' from s_j, i.e., $\mathcal{L}(M|_s) \supseteq \mathcal{L}(M'|_{s_j})$.

By the definition of $\mathcal{L}(M') \subseteq \mathcal{L}(M)$, $\forall s_j \in R'$ $(1 \leq j \leq n)$ $\exists r \in R$ such that $\mathcal{L}(M|_r) \supseteq \mathcal{L}(M'|_{s_j})$. By construction of K, $r \in c_j$. Therefore K is a cover of M.

We now show K is closed, i.e., $\forall i \in I$ $\forall c_j \in K$ $\exists o \in O$ $\{$ $[(\forall s \in c_j$ $\Lambda(i,s,o) = 1] \cdot [\exists c_k \in K$ $\forall s \in c_j$ $\delta(i,s,o) \in c_k)]$ $\}$ according to Lemma 2.12. Consider an arbitrary input $i \in I$ and an arbitrary element $c_j \in K$. As M' is a DFSM, there exists a unique $o \in O$ such that $T'(i, s_j, s'_j, o) = 1$ for some s'_j. Now let us concentrate on this particular o. By the definition of K, we know for every $s \in c_j$, $\mathcal{L}(M|_s) \subseteq \mathcal{L}(M'|_{s_j})$. The trace (i,o) is in $\mathcal{L}(M'|_{s_j})$, and therefore is also in $\mathcal{L}(M|_s)$. So $\forall s \in c_j$ $\Lambda(i,s,o) = 1$. We now prove the last term in the conjunction. Assume the last term is false, i.e., $\forall c_k \in K$ $\exists s \in c_j$ $\delta(i,s,o) \notin c_k$. For every $c_k \in K$, there exists a pair of states $\{\delta(i,p_k,o), \delta(i,q_k,o)\} \not\subseteq c_k$, where p_k and q_k are elements of c_j. Consider the state s_k associated to c_k, then $\mathcal{L}(M'|_{s_k})$ is not contained in both $\mathcal{L}(M|_{\delta(i,p_k,o)})$ and $\mathcal{L}(M|_{\delta(i,q_k,o)})$, since $\{\delta(i,p_k,o), \delta(i,q_k,o)\} \not\subseteq c_k$. As M' is deterministic, at state s_k, on some input sequence $i_1, i_2, \ldots, i_p$, there must be a corresponding output sequence of M' that cannot be produced by either $\delta(i,p_k,o)$ or $\delta(i,q_k,o)$, i.e., these next states cannot produce the same output in M. Then on input sequence $i, i_1, i_2, \ldots, i_p$, p_k and q_k cannot produce the same output in M. This implies that p_k and q_k are not included in any $c_j : 1 \leq j \leq n$, which contradicts the assumption and proves $\exists c_k \in K$ $\forall s \in c_j$ $\delta(i,s,o) \in c_k$. In summary, for arbitrary i and c_j, the formula $\exists o \in O$ $\{$ $[(\forall s \in c_j$ $\Lambda(i,s,o) = 1] \cdot [\exists c_k \in K$ $\forall s \in c_j$ $\delta(i,s,o) \in c_k)]$ $\}$ is true. Therefore K is closed. ■

Note in Theorem 2.15, M cannot be easily generalized to an NDFSM, because the above proof would not go through.

The following is the main theorem proving the correctness of exact state minimization algorithms which are based on closed covers. It shows that one can explore all behaviors contained within a PNDFSM by finding all closed covers for the PNDFSM.

Theorem 2.16 *Let M be a PNDFSM and M' be a DFSM. $\mathcal{L}(M') \subseteq \mathcal{L}(M)$ if and only if there exists a closed cover for M which is represented by M'.*

Proof: The *only if* part immediately follows from Theorem 2.15. The *if* part immediately follows from Theorem 2.14. ■

2.7 STATE MINIMIZATION PROBLEMS

Definition 2.29 *Given an NDFSM* $M = \langle S, I, O, T, R \rangle$, *the* **state minimization problem** *is to find a DFSM* $M' = \langle S', I, O, T', R' \rangle$ *such that*

1. $\mathcal{L}(M') \subseteq \mathcal{L}(M)$, *and*
2. $\forall M''$ *such that* $\mathcal{L}(M'') \subseteq \mathcal{L}(M)$, $|S'| \leq |S''|$. [11]

Such a case is denoted by $\mathcal{L}(M') \overset{min}{\subseteq} \mathcal{L}(M)$.

M' is not required to be deterministic. On the contrary, to preserve flexibility for other sequential synthesis steps, one may extend the definition and choose the M' with maximal non-determinism in specification, out of all state minimum machines.

Definition 2.23 of behavioral containment and Definition 2.29 of the state minimization problem apply also to other kinds of FSMs, which are subclasses of PNDFSMs.

Definition 2.30 *A closed cover* K *for* M *is called a* **minimum closed cover** *for* M *if every closed cover* K' *of* M *has* $|K'| \geq |K|$.

The following theorem is a companion and an extension of Theorem 2.16. It proves the optimality of exact state minimization algorithms which find minimum closed covers.

Theorem 2.17 *Let* M *be a PNDFSM and* M' *be a DFSM.* $\mathcal{L}(M') \overset{min}{\subseteq} \mathcal{L}(M)$ *if and only if there exists a minimum closed cover for* M *which is represented by* M'.

[11] Given a set S, $|S|$ denotes the cardinality of the set.

Proof: Assume $\mathcal{L}(M') \overset{min}{\subseteq} \mathcal{L}(M)$. By Theorem 2.15, there exists a closed cover K for M which is represented by M'. Suppose there exists a closed cover K'' for M such that $|K''| < |K|$. By Theorem 2.14, there exists a DFSM M'' representing K'' such that $\mathcal{L}(M'') \subseteq \mathcal{L}(M)$. This contradicts with our initial assumption so K must be a minimum closed cover for M.

Assume K is a minimum closed cover for M. By Theorem 2.14, there exists a DFSM M' representing K and $\mathcal{L}(M') \subseteq \mathcal{L}(M)$. If $\mathcal{L}(M'') \overset{min}{\subseteq} \mathcal{L}(M)$, there exists a minimum closed cover K'' for M such that K'' represents M''. However K must have the same number of sets as K'' since K is also a minimum closed cover for M. Thus M' has the same number of states as M'' so that $\mathcal{L}(M') \overset{min}{\subseteq} \mathcal{L}(M)$. ■

The state minimization problem defined above is very different from the NDFA minimization problem described in classical automata textbooks. Here we want a minimal state implementation which is contained in the specification, while the classical problem is defined as:

Definition 2.31 *Given an NDFSM M, the* **behavior-preserving state minimization problem** *is to find an NDFSM M' which represents the same set of behaviors as M but has the fewest number of states.*

The above notion of behavior-preserving state minimization may be useful in formal verification because all behaviors specified must be verified and thus must be *preserved* during minimization.

2.8 POWER NDFSM AND STG-CONTAINMENT

Now let us revisit the question: Can we explore behaviorally contained DFSMs by STG-containment? This is an interesting question because intuitively, STG-containment is easier to compute than behavioral containment. In this section, we shall show that given a PNDFSM M, we can derive a new NDFSM called M^{power} such that M' is behaviorally contained in M if and only if M' is STG-contained in M^{power}.

Definition 2.32 *Given a PNDFSM* $M = \langle S, I, O, T, R \rangle$, *the* **power NDFSM** *is* $M^{power} = \langle 2^S, I, O, T^{power}, R^{power} \rangle$. T^{power} *is a transition relation in* $I \times 2^S \times 2^S \times O$. $T^{power}(i, c_j, c_k, o) = 1$ *if and only if* $\forall s \in c_j \; \exists s' \in c_k \; T(i, s, s', o) = 1$. $R^{power} \subseteq 2^S$ *includes every state set which contains a reset state* $r \in R$.

Note that M^{power} is not pseudo non-deterministic because given any transition $(i, c_j, c_k, o) \in T^{power}$, for every $\hat{c_k} \supset c_k$, there is another transition such that $T^{power}(i, c_j, \hat{c_k}, o) = 1$ and $R^{power}(c) = 1$.

The power machine is basically a representation in FSM format of all possible collections of states and transitions between these collections. When restricted to compatibles, as in the next definition, the power machine is just a different way of looking to all possible covers of compatibles.

Definition 2.33 *Given a power NDFSM* $M^{power} = \langle 2^S, I, O, T^{power}, R^{power} \rangle$, *the* **power NDFSM with its state space restricted to a subset** K *of state sets, denoted by* $M^{power}|_K$, *is the NDFSM* $\langle K, I, O, T^{power}|_K, R^{power}|_K \rangle$. $T^{power}|_K(i, c, c', o) = 1$ *if and only if* $c, c' \in K$ *and* $T^{power}(i, c, c', o) = 1$. $R^{power}|_K(c) = 1$ *if and only if* $c \in K$ *and* $R^{power}(c) = 1$.

Lemma 2.18 K *is a closed cover if*

1. $K \cap R^{power} \neq \emptyset$, *and*
2. *for every* $i \in I$ *and* $c_j \in K$, *there exists* $c_k \in K$ *and* $o \in O$ *such that* $T^{power}(i, c_j, c_k, o) = 1$.

Proof: Condition 1 of Lemma 2.18 is true if and only if there exists c_j such that $c_j \in R^{power}$ and $c_j \in K$. By Definition 2.32, $c_j \in R^{power}$ implies that there exists a reset state $r \in R$ such that $r \in c_j$. As $c_j \in K$, K is a cover according to Definition 2.25. By substituting the definition of T^{power} into condition 2 of Lemma 2.18, the closure condition as expressed by Definition 2.26 is satisfied. ■

This lemma suggests an alternative way of finding a closed cover, K. Looking to the power PNDFSM M^{power}, K corresponds to a set of states in 2^S such that (1) at least one state in K is a reset state in R^{power}, and (2) for each input i, each state in K can have a next state also in K under transitions in T^{power}. This selection procedure is analogous to finding a DFSM which is STG-contained within the power PNDFSM, and covers at least one reset state.

Lemma 2.19 *K is represented by M' if the power NDFSM with its state space restricted to K STG-contains M', i.e., $STG(M^{power}|_K) \supseteq STG(M')$.*

Proof: By Definition 2.28, K is represented by M' if and only if for every $i \in I$ and $j : 1 \leq j \leq n$, there exists $k : 1 \leq k \leq n$ and $o \in O$ such that, if $T'(i, s_j, s_k, o) = 1$ then $T^{power}(i, c_j, c_k, o) = 1$. Let $M^{power}|_K$ be the power NDFSM of M with its state space restricted to the set K. By Definition 2.24, K is represented by M' if and only if $STG(M^{power}|_K) \supseteq STG(M')$. ■

Now we are ready to extend Theorem 2.16 and provide another way of exploring behaviors contained within a PNDFSM, beside enumerating closed covers.

Theorem 2.20 *Let M be a PNDFSM and M' be a DFSM. The following statements are equivalent:*

1. $\mathcal{L}(M') \subseteq \mathcal{L}(M)$.
2. *There exists a closed cover for M which is represented by M'.*
3. *There exists a subset $K \subseteq 2^S$ such that*
   - $K \cap R^{power} \neq \emptyset$, *and*
   - $\forall i \in I \; \forall c_j \in K \; \exists c_k \in K \; \exists o \in O \; T^{power}(i, c_j, c_k, o) = 1$, *and*
   - $STG(M') \subseteq STG(M^{power}|_K)$.

Proof: Equivalence of statements 1 and 2 follows from Theorem 2.16. By applying Lemmas 2.18 and 2.19 and gathering their conditions, statement 2 becomes statement 3. ■

Definition 2.34 *A DFSM M' is a* **minimum STG-contained DFSM in a PNDFSM** *M, denoted by $STG(M') \overset{min}{\subseteq} STG(M)$, if*

1. $STG(M') \subseteq STG(M)$, *and*
2. $\forall M''$ *such that* $STG(M'') \subseteq STG(M)$, $|S'| \leq |S''|$.

Theorem 2.21 *Let M be a PNDFSM and M' be a DFSM. The following statements are equivalent:*

1. $\mathcal{L}(M') \overset{min}{\subseteq} \mathcal{L}(M)$.
2. *There exists a minimum closed cover for* M *which is represented by* M'.
3. *There exists a subset* $K \subseteq 2^S$ *such that*
   - $K \cap R^{power} \neq \emptyset$, *and*
   - $\forall i \in I\ \forall c_j \in K\ \exists c_k \in K\ \exists o \in O\ T^{power}(i, c_j, c_k, o) = 1$, *and*
   - $STG(M') \overset{min}{\subseteq} STG(M^{power}|_K)$.

Theorem 2.21 summarizes succinctly the theory we know about state minimization.

Although we have the theory, we do not have yet an exact algorithm for state minimization which explores STG-contained DFSMs within the power NDFSM. However, this is not due to an inability to represent M^{power} compactly as it can be represented implicitly as shown in Chapter 7. In Chapter 8, we will describe a heuristic procedure to find a minimal DFSM STG-contained in the power NDFSM. In the next section, we'll outline the common steps used in state minimization algorithms which are based on closed cover, for different subclasses of NDFSMs.

2.9 EXACT ALGORITHM FOR STATE MINIMIZATION

By Theorem 2.17, the state minimization problem of a PNDFSM can be reduced to the problem of finding a minimum closed cover for the PNDFSM. From what we know so far, a closed cover may contain any arbitrary set of states. Once such a minimum closed cover is found, a minimum state DFSM which represents the closed cover can be obtained easily, and this final step is traditionally called **mapping**. A brute force approach to find a minimum closed cover would be to enumerate sets of state sets, and test each one of them to see if it represents a closed cover according to Definition 2.25 and Lemma 2.12. Out of all the closed covers, one would pick one of minimum cardinality. This section will discuss ways to improve on this brute force algorithm.

In the exact method described above, each candidate closed cover includes subsets of 2^S where S is the state space of the PNDFSM. The number of such

state sets to be generated is exponential in the number of states, $|S|$. Actually we do not have to consider all subsets, but only those that may be in a closed cover. The definition of a closed cover requires that it contains subsets only of the following types.

Definition 2.35 *A set of states is an* **output compatible** *if for every input, there is a corresponding output which can be produced by each state in the set.*

Lemma 2.22 *Every element of a closed cover* [12] *is an output compatible.*

Proof: By Definition 2.26, for each set $c_j : 1 \leq j \leq n$ on every input $i \in I$, there is an output $o \in O$ that can be produced by each state in the set. Therefore c_j is an output compatible. ■

Definition 2.36 *A set of states is a* **compatible** *if for each input sequence, there is a corresponding output sequence which can be produced by each state in the compatible.*

Lemma 2.23 *Every element of a closed cover is a compatible.*

Proof: The lemma can be proved by showing that if K is a closed set of output compatibles then K is a closed set of compatibles. (The converse is trivially true because a compatible is also an output compatible.) Assume K is a closed set of output compatibles but there exists $c_j \in K$ such that $p_1 \in c_j$ and $q_1 \in c_j$ are not compatible. Thus there exists some input sequence $i_1, i_2, \ldots, i_k$ to M which produces state sequences $p_1, p_2, \ldots, p_k$ and $q_1, q_2, \ldots, q_k$, respectively. And there is no output $o \in O$ such that $T(i, p_k, p_{k+1}, o) = 1$ and $T(i, q_k, q_{k+1}, o) = 1$, for any p_{k+1} and q_{k+1}. As K is closed, for any $m : 1 \leq m \leq n$, $\{p_m, q_m\}$ is contained in some set in K. However $\{p_k, q_k\}$ contradicts our assumption that each set in K is output compatible and proves that each $c_j \in K$ is a compatible. ■

Because of Lemma 2.23, an exact state minimization algorithm only needs to generate compatibles. The next step of an exact algorithm after compatible generation is to select a subset of compatibles that corresponds to a minimized

[12] This more restrictive lemma is also true: Every element of a closed *set* is an output compatible.

machine. To satisfy behavioral containment, the selection of compatibles should be such that appropriate covering and closure conditions are met. The covering conditions guarantee that some selected compatible (i.e., some state in the minimized machine) corresponds to a reset state of the original machine. The closure conditions require that for each selected compatible, the compatibles implied by state transitions should also be selected. The state minimization problem reduces to one that selects a minimum closed cover of compatibles. Instead of enumerating and testing all subset of compatibles, the selection is usually solved as a **binate covering problem**. Covering and closure conditions are expressed as binate clauses, and the minimum solution can be searched in a systematic manner.

The set of compatibles is still very large. For the purpose of state minimization, it is sufficient to identify a minimum subset called the prime compatibles such that a minimum closed cover of prime compatibles still yields a minimum behaviorally contained machine. Compatibles that are not *dominated* by other compatibles are called prime compatibles.

Definition 2.37 *A compatible c'* **prime dominates** *a compatible c if for each minimum closed cover containing c, the selection with c replaced by c' also corresponds to a minimum closed cover.*

Definition 2.38 *A compatible c is a* **prime compatible** *if there does not exists another compatible c' such that c' prime dominates c.*

Theorem 2.24 *There exists a minimum closed cover made up entirely of prime compatibles.*

Proof: Suppose C is a minimum closed cover which contains a compatible c that is not prime, i.e., there exists a prime compatible $\hat{c}$ which dominates c. By Definition 2.38, the set $(C - \{c\}) \cup \{\hat{c}\}$ must be another minimum closed cover with one more prime compatible. One can continue this substitution process until the minimum closed cover is made up entirely of prime compatibles. ■

The actual computations of compatibles and primeness differ for different types of FSMs. Also the related covering problems vary slightly. Part II of this book will discuss the compatible computation and covering for ISFSMs. The corresponding problems for PNDFSMs are addressed in Part III of this book.