Chapter 9

Algorithms and Decision Procedures for Regular Languages
Decision Procedures

A decision procedure is an algorithm whose result is a Boolean value. It must:

- Halt
- Be correct

Important decision procedures exist for regular languages:

- Given an FSM $M$ and a string $s$, does $M$ accept $s$?
- Given a regular expression $\alpha$ and a string $w$, does $\alpha$ generate $w$?
Membership

We can answer the membership question by running an FSM.

But we must be careful:
Membership

\[ \text{decideFSM}(M: \text{FSM}, \; w: \text{string}) = \]
\[ \text{If } \text{ndfsmsimulate}(M, \; w) \text{ accepts then return True else return False.} \]

\[ \text{decideregex}(\alpha: \text{regular expression}, \; w: \text{string}) = \]
\[ \text{From } \alpha, \text{ use regextofsm to construct an FSM } M \]
\[ \text{such that } L(\alpha) = L(M). \]
\[ \text{Return decideFSM}(M, \; w). \]
Emptiness and Finiteness

- Given an FSM $M$, is $L(M)$ empty?
- Given an FSM $M$, is $L(M) = \Sigma_M^*$?
- Given an FSM $M$, is $L(M)$ finite?
- Given an FSM $M$, is $L(M)$ infinite?
- Given two FSMs $M_1$ and $M_2$, are they equivalent?
Emptiness

• Given an FSM $M$, is $L(M)$ empty?

• The graph analysis approach:

• The simulation approach:
Emptiness

- Given an FSM $M$, is $L(M)$ empty?

- The graph analysis approach:
  1. Mark all states that are reachable via some path from the start state of $M$.
  2. If at least one marked state is an accepting state, return $False$. Else return $True$.

- The simulation approach:
Emptiness

• Given an FSM $M$, is $L(M)$ empty?

• The graph analysis approach:
  1. Mark all states that are reachable via some path from the start state of $M$.
  2. If at least one marked state is an accepting state, return $False$. Else return $True$.

• The simulation approach:
  1. Let $M' = ndfsmtodfsm(M)$.
  2. For each string $w$ in $\Sigma^*$ such that $|w| < |K_M|$ do:
     Run $decideFSM(M', w)$.
  3. If $M'$ accepts at least one such string, return $False$. Else return $True$. 
Totality

- Given an FSM $M$, is $L(M) = \Sigma_M^*$?

1. Construct $M'$ to accept $\neg L(M)$.
2. Return $emptyFSM(M')$. 
Finiteness

• Given an FSM $M$, is $L(M)$ finite?

• The graph analysis approach:
Finiteness

• Given an FSM $M$, is $L(M)$ finite?

• The graph analysis approach:

  The mere presence of a loop does not guarantee that $L(M)$ is infinite. The loop might be:

  • labeled only with $\varepsilon$,
  • unreachable from the start state, or
  • not on a path to an accepting state.
Finiteness

• Given an FSM $M$, is $L(M)$ finite?

• The graph analysis approach:

  1. $M' = ndfsmtodfsm(M)$.
  2. $M'' = minDFSM(M')$.
  3. Mark all states in $M''$ that are on a path to an accepting state.
  4. Considering only marked states, determine whether there are any cycles in $M''$.
  5. If there are cycles, return $True$. Else return $False$. 
Finiteness

• Given an FSM $M$, is $L(M)$ finite?

• The simulation approach:

1. $M' = ndfsmtodfsm(M)$.
2. For each string $w$ in $\Sigma^*$ such that _________________ do:
   
   Run $\text{decideFSM}(M', w)$.
3. If $M'$ accepts at least one such string, return $\text{False}$.
   Else return $\text{True}$. 
Finiteness

- Given an FSM $M$, is $L(M)$ finite?

- The simulation approach:

  1. $M' = ndfsmtodfsm(M)$.
  2. For each string $w$ in $\Sigma^*$ such that $|K_M'| \leq w \leq 2 \cdot |K_M'| - 1$ do:
     - Run $\text{decideFSM}(M', w)$.
  3. If $M'$ accepts at least one such string, return $False$. Else return $True$. 
Equivalence

- Given two FSMs $M_1$ and $M_2$, are they equivalent? In other words, is $L(M_1) = L(M_2)$? We can describe two different algorithms for answering this question.
Equivalence

- Given two FSMs $M_1$ and $M_2$, are they equivalent? In other words, is $L(M_1) = L(M_2)$?

$equalFSMs_1(M_1: FSM, M_2: FSM) =$
1. $M_1' = buildFSMcanonicalform(M_1)$.
2. $M_2' = buildFSMcanonicalform(M_2)$.
3. If $M_1'$ and $M_2'$ are equal, return $True$, else return $False$. 
Equivalence

- Given two FSMs $M_1$ and $M_2$, are they equivalent? In other words, is $L(M_1) = L(M_2)$?

Observe that $M_1$ and $M_2$ are equivalent iff:

$$(L(M_1) - L(M_2)) \cup (L(M_2) - L(M_1)) = \emptyset.$$ 

$equalFSMs_2(M_1: \text{FSM}, M_2: \text{FSM}) =$

1. Construct $M_A$ to accept $L(M_1) - L(M_2)$.
2. Construct $M_B$ to accept $L(M_2) - L(M_1)$.
3. Construct $M_C$ to accept $L(M_A) \cup L(M_B)$.
4. Return $emptyFSM(M_C)$. 
Minimality

- Given DFSM $M$, is $M$ minimal?
Minimality

• Given DFSM $M$, is $M$ minimal?

1. $M' = \text{minDFSM}(M)$.
2. If $|K_M| = |K_{M'}|$ return True; else return False.
Answering Specific Questions

Given two regular expressions $\alpha_1$ and $\alpha_2$, is:

$$(L(\alpha_1) \cap L(\alpha_2)) - \{\varepsilon\} \neq \emptyset?$$
Answering Specific Questions

Given two regular expressions $\alpha_1$ and $\alpha_2$, is:

$$(L(\alpha_1) \cap L(\alpha_2)) - \{\varepsilon\} \neq \emptyset?$$

1. From $\alpha_1$, construct an FSM $M_1$ such that $L(\alpha_1) = L(M_1)$.
2. From $\alpha_2$, construct an FSM $M_2$ such that $L(\alpha_2) = L(M_2)$.
3. Construct $M'$ such that $L(M') = L(M_1) \cap L(M_2)$.
4. Construct $M_\varepsilon$ such that $L(M_\varepsilon) = \{\varepsilon\}$.
5. Construct $M''$ such that $L(M'') = L(M') - L(M_\varepsilon)$.
6. If $L(M'')$ is empty return $False$; else return $True$. 
Answering Specific Questions

Given two regular expressions $\alpha_1$ and $\alpha_2$, are there at least 3 strings that are generated by both of them?
Summary of Algorithms

- Operate on FSMs without altering the language that is accepted:
  - \textit{Ndfsmtodfs}
  - \textit{MinDFSM}
Summary of Algorithms

- Compute functions of languages defined as FSMs:
  - Given FSMs $M_1$ and $M_2$, construct a FSM $M_3$ such that $L(M_3) = L(M_2) \cup L(M_1)$.
  - Given FSMs $M_1$ and $M_2$, construct a new FSM $M_3$ such that $L(M_3) = L(M_2) \cdot L(M_1)$.
  - Given FSM $M$, construct an FSM $M^*$ such that $L(M^*) = (L(M))^*$.
  - Given a DFSM $M$, construct an FSM $M^*$ such that $L(M^*) = \neg L(M)$.
  - Given two FSMs $M_1$ and $M_2$, construct an FSM $M_3$ such that $L(M_3) = L(M_2) \cap L(M_1)$.
  - Given two FSMs $M_1$ and $M_2$, construct an FSM $M_3$ such that $L(M_3) = L(M_2) - L(M_1)$.
  - Given an FSM $M$, construct an FSM $M^*$ such that $L(M^*) = (L(M))^R$.
  - Given an FSM $M$, construct an FSM $M^*$ that accepts $letsu(L(M))$. 
Algorithms, Continued

● Converting between FSMs and regular expressions:
  ● Given a regular expression $\alpha$, construct an FSM $M$ such that:
    \[ L(\alpha) = L(M) \]
  
  ● Given an FSM $M$, construct a regular expression $\alpha$ such that:
    \[ L(\alpha) = L(M) \]

● Algorithms that implement operations on languages defined by regular expressions: any operation that can be performed on languages defined by FSMs can be implemented by converting all regular expressions to equivalent FSMs and then executing the appropriate FSM algorithm.
Algorithms, Continued

- Converting between FSMs and regular grammars:
  - Given a regular grammar $G$, construct an FSM $M$ such that:
    \[ L(G) = L(M) \]
  - Given an FSM $M$, construct a regular grammar $G$ such that:
    \[ L(G) = L(M). \]
Algorithms: Decision Procedures

- Decision procedures that answer questions about languages defined by FSMs:
  - Given an FSM $M$ and a string $s$, decide whether $s$ is accepted by $M$.
  - Given an FSM $M$, decide whether $L(M)$ is empty.
  - Given an FSM $M$, decide whether $L(M)$ is finite.
  - Given two FSMs, $M_1$ and $M_2$, decide whether $L(M_1) = L(M_2)$.
  - Given an FSM $M$, is $M$ minimal?

- Decision procedures that answer questions about languages defined by regular expressions: Again, convert the regular expressions to FSMs and apply the FSM algorithms.