CSE 105: Project Task 2

Due on Feb 22, 2024 at 5:00pm

Professor Minnes Kemp

Ray Tsai

A16848188

Context

In the context of UCSD's course *Theory of Computability* (CSE 105), Task 2 challenges students to illustrate their understanding of deterministic finite automata (DFA) and Turing machines through practical application (see here). Students are required to construct a DFA to recognize a specific pattern and then using that DFA as a basis to design a Turing machine that proves that the language encoding this pattern is decidable. This demonstrates decidability by establishing a Turing machine that can methodically evaluate the submission against the project's criteria.

For a submission to be considered valid, it must follow:

Step 1 - Give the clear **context** and justification for the chosen application.

Step 2 - Specify the specified **alphabet** and precise description of the **language** relevant to the application.

Step 3 - Give **Examples** of strings within and outside the set, with explanations.

Step 4 - Design a **DFA** that recognizes this language, with a state diagram and justification of its functionality.

Step 5 - Construct a **Turing machine** based on the DFA, with additional states as required.

Step 6 - Give an illustration of the Turing machine's **computation** on a selected string from the DFA.

In this task, we will design a DFA that assess whether a submission adheres to the task's specified guidelines. Subsequently, this DFA will serve as the foundation for constructing a Turing machine, which will establish that the language encoding the chosen pattern is indeed decidable. If the submission correctly fulfills all criteria, the Turing machine will accept it; if not, it will be rejected. The choice of this application is due to its direct relation to a project in the *Theory of Computability* (CSE 105) course that I am currently enrolled in and **care about a lot**.

Alphabet and Language

For this application, we will use the alphabet $\Sigma = \{(CON), (AL), (EX), (DFA), (TM), (COMP)\}$, with each symbol representing an abbreviation for the initial letters of the key components in each task step: (CON) for context, (AL) for alphabet and language, (EX) for example, (DFA) for DFA, (TM) for Turing machine, and $(COMP)$ for computation.

The language encoding our chosen pattern is described by the regular expression

$$R = (CON)^+(AL)(EX)^2(DFA)(TM)(COMP).$$

The expression R specifies a sequence of setps where each of the six steps must appear at least once and in the correct order. The use of the plus sign $(+)$ after (CON) indicates that these steps can be repeated one or more times, reflecting their potential multiple occurrences (paragraphs) in a project submission.

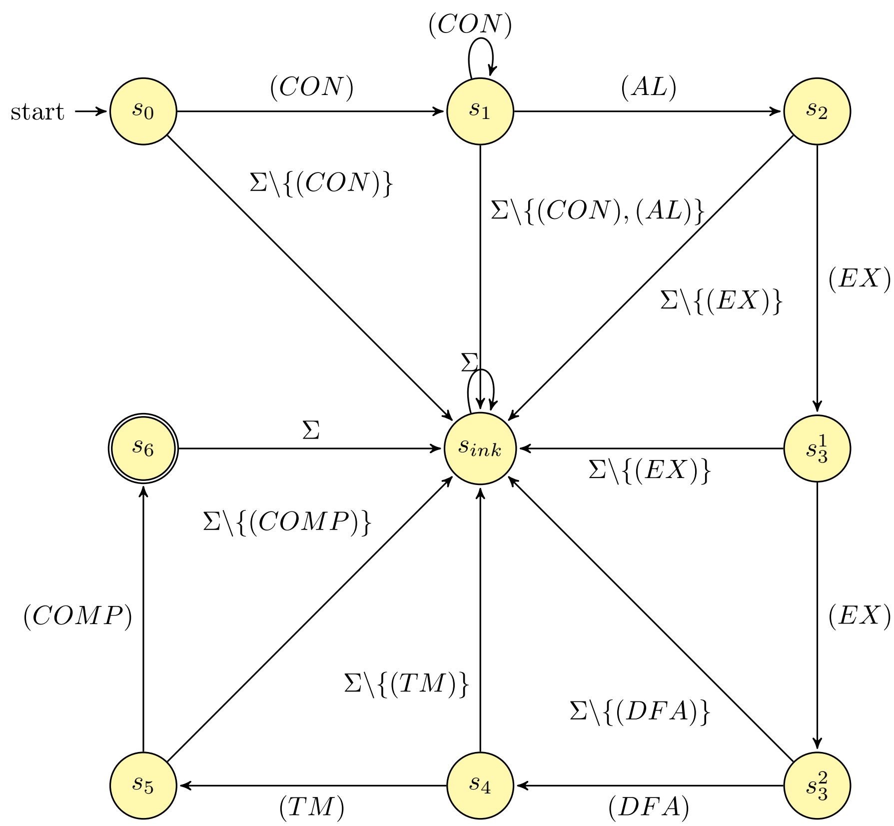
Example^2

An example within $L(R)$ is $w = (CON)(AL)(EX)(EX)(DFA)(TM)(COMP)$, representing the minimal sequence for a valid submission, containing all essential components in the correct order. Conversely, $s = (CON)(AL)(EX)(EX)$ is not in $L(R)$. The choice of s is due to its reflection of our current progress. Since we have not completed the task, s illustrates an invalid submission, and thus $s \notin L(R)$.

DFA

We now design a DFA M that recognizes $L(R)$.

State diagram of M

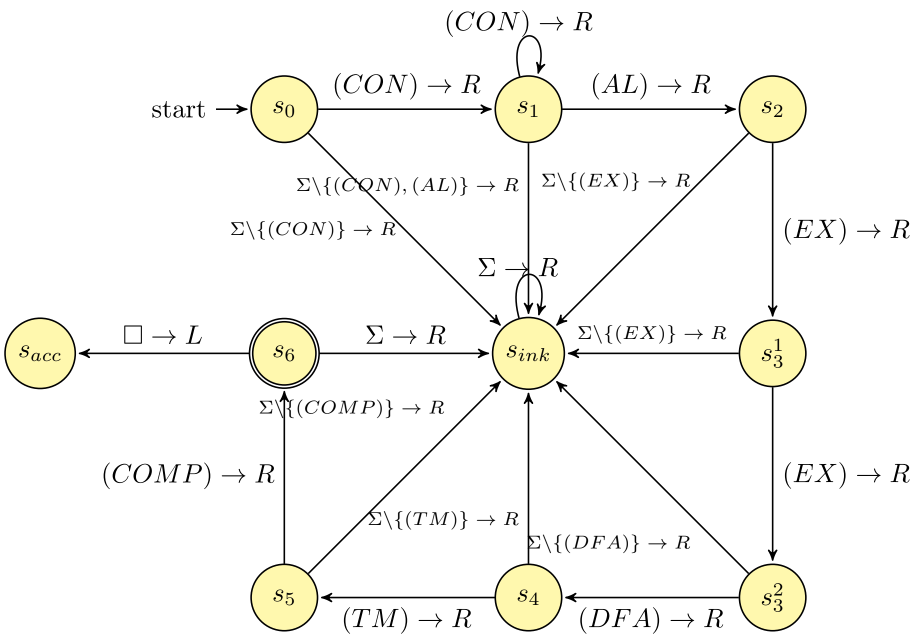


The DFA M is designed with a set of states $Q = \{s_0, s_1, s_2, s_3^1, s_3^2, s_4, s_5, s_6, s_{ink}\}$. Each state s_i corresponds to a specific step in the validation sequence of the input string, with s_0 as the initial state. Note that s_3^1, s_3^2 represents the number of (EX) recieved from the input string. In addition, the s_{ink} state is a non-accepting state that absorbs any string deviating from the language's specified sequence. Transitions from each state to the s_{ink} state occur upon reading an input not expected at that stage, ensuring that only strings following the exact order of components $(CON)^+(AL)(EX)^2(DFA)(TM)(COMP)$ are accepted.

See next page for the construciton of a Turing machine which decides $L(R)$.

Turing Machine

Here is the resulting T following the construction in the proof of Theorem 4.1:



Note that, additional to Q, T also contains s_{acc} and s_{rej}. However, we have omitted the appearance of s_{rej} and all edges associated with it from the daigram, due to the convention.

Computation

We show the computation of string w from Step 3 in T. Note that w reflects our current progress of task 2.

$s_0 \downarrow$							
(CON)	(AL)	(EX)	(EX)	(DFA)	(TM)	$(COMP)$	$\square$
	$s_1 \downarrow$						
(CON)	(AL)	(EX)	(EX)	(DFA)	(TM)	$(COMP)$	$\square$
		$s_2 \downarrow$					
(CON)	(AL)	(EX)	(EX)	(DFA)	(TM)	$(COMP)$	$\square$
			$s_3^1 \downarrow$				
(CON)	(AL)	(EX)	(EX)	(DFA)	(TM)	$(COMP)$	$\square$
				$s_3^2 \downarrow$			
(CON)	(AL)	(EX)	(EX)	(DFA)	(TM)	$(COMP)$	$\square$
					$s_4 \downarrow$		
(CON)	(AL)	(EX)	(EX)	(DFA)	(TM)	$(COMP)$	$\square$
						$s_5 \downarrow$	
(CON)	(AL)	(EX)	(EX)	(DFA)	(TM)	$(COMP)$	$\square$
							$s_6 \downarrow$
(CON)	(AL)	(EX)	(EX)	(DFA)	(TM)	$(COMP)$	$\square$
						$s_{acc} \downarrow$	
(CON)	(AL)	(EX)	(EX)	(DFA)	(TM)	$(COMP)$	$\square$