

Computability Crib Sheet

Solving problems related to decidability and undecidability requires that you know and be able to exploit various facts that we have seen in class and are in the text. Many of these pertain to properties of regular and context-free languages. Here we will summarize the facts and point you to examples. If you can't understand what is stated here, or, having looked at the solutions of the example problems, don't understand why they work, you have a lot of studying to do. On the quizzes and exams you will be expected to have these facts at your fingertips and be able to exploit them effectively. (For the final, all this is in the syllabus. For Quiz 3, whatever part we have covered by then.)

There are three classes of facts, as described below.

1 Computable transforms

We have seen that the classes of regular and context-free languages possess numerous closure properties. The proofs of these closure properties, as per our template, are constructive. For example, to show that the class of regular languages is closed under complement, we take a DFA D_1 and show how to obtain from it a DFA D_2 such that $L(D_2)$ is the complement of $L(D_1)$. In this case, transforming D_1 into D_2 is easy: we only change the set of final states, setting the set of final states of D_2 to the set of non-final states of D_1 . Notice that this simple transformation is easily automated, in the sense that we can write a program (and thus by the Church-Turing thesis a TM) that on input $\langle D_1 \rangle$ returns $\langle D_2 \rangle$. What you need to remember is that this TM exists. You will need to use it as a subroutine in solving problems on the final.

It turns out that this type of property is true for all closure properties that we have seen. The table of Figure 1 summarizes the different resulting TMs that can be built and which you should assume are available to you. In the exam, refer to them by name in your solutions.

Make sure you understand what these TMs accomplish, in the sense of the property they have.

These are not the only such transforms. There are many others, underlying other closure properties that we have seen.

2 Decidable languages

The second set of facts that you need to remember and be able to use is that certain specific languages that we have seen are decidable. Below, we will list these languages. For each language, spend time to look at the definition of the language and understand what it describes. Then, go

TM	Input	Output	Property
$T_{\text{dfa}}^{\text{comp}}$	$\langle D_1 \rangle$ where D_1 is DFA	$\langle D_2 \rangle$ where D_2 is a DFA	$L(D_2) = \overline{L(D_1)}$
T_{dfa}^{\cap}	$\langle D_1, D_2 \rangle$ where D_1, D_2 are DFAs	$\langle D \rangle$ where D is a DFA	$L(D) = L(D_1) \cap L(D_2)$
T_{dfa}^{\cup}	$\langle D_1, D_2 \rangle$ where D_1, D_2 are DFAs	$\langle D \rangle$ where D is a DFA	$L(D) = L(D_1) \cup L(D_2)$
T_{dfa}^{\bullet}	$\langle D_1, D_2 \rangle$ where D_1, D_2 are DFAs	$\langle D \rangle$ where D is a DFA	$L(D) = L(D_1) \cdot L(D_2)$
T_{dfa}^*	$\langle D_1 \rangle$ where D_1 is a DFA	$\langle D_2 \rangle$ where D_2 is a DFA	$L(D_2) = L(D_1)^*$
$T_{\text{nfa} \rightarrow \text{dfa}}$	$\langle N \rangle$ where N is a NFA	$\langle D \rangle$ where D is a DFA	$L(D) = L(N)$
T_{cfg}^{\cup}	$\langle G_1, G_2 \rangle$ where G_1, G_2 are CFGs	$\langle G \rangle$ where G is a CFG	$L(G) = L(G_1) \cup L(G_2)$
T_{cfg}^{\bullet}	$\langle G_1, G_2 \rangle$ where G_1, G_2 are CFGs	$\langle G \rangle$ where G is a CFG	$L(G) = L(G_1) \cdot L(G_2)$
T_{cfg}^*	$\langle G_1 \rangle$ where G_1 is a CFG	$\langle G_2 \rangle$ where G_2 is a CFG	$L(G_2) = L(G_1)^*$
$T_{\text{dfa} \rightarrow \text{cfg}}$	$\langle D \rangle$ where D is a DFA	$\langle G \rangle$ where G is a CFG	$L(G) = L(D)$
$T_{\text{cfg} \rightarrow \text{cnf}}$	$\langle G \rangle$ where G is a CFG	$\langle G' \rangle$ where G' is a CFG in CNF	$L(G) = L(G')$

Figure 1: Transform-computing TMs to know. CNF stands for Chomsky Normal Form.

find in the text the theorem that says it is decidable, and make sure you understand what this claim means.

Your solutions to problems may use the TMs given in Figure 2 as subroutines in the TMs you design. For each machine, we state the language it decides.

These are not the only decidable languages we have seen. There are many others as well.

2.1 Undecidable languages

The third set of facts that you need to remember and be able to use is that certain specific languages that we have seen are undecidable. (That is, they are *not* decidable.) Below, we will list these languages. For each language, spend time to look at the definition of the language and understand what it describes. Then, go find in the text the theorem that says it is undecidable, and make sure you understand what this claim means.

The following languages are undecidable. In your solutions on the exam, refer to these languages by name:

$$\begin{aligned}
 \text{HALT}_{\text{TM}} &= \{ \langle M, w \rangle : M \text{ is a TM and } M(w) \text{ halts} \} \\
 A_{\text{TM}} &= \{ \langle M, w \rangle : M \text{ is a TM and } M(w) \text{ accepts} \} \\
 E_{\text{TM}} &= \{ \langle M \rangle : M \text{ is a TM and } L(M) = \emptyset \} \\
 \text{ALL}_{\text{CFG}} &= \{ \langle G \rangle : G \text{ is a CFG and } L(G) = \Sigma^* \}
 \end{aligned}$$

These are not the only undecidable languages we have seen. There are many others as well.

TM	Language that it decides
M_{dfa}	$A_{\text{DFA}} = \{ \langle D, w \rangle : D \text{ is a DFA and } D \text{ accepts } w \}$
M_{nfa}	$A_{\text{NFA}} = \{ \langle N, w \rangle : N \text{ is a NFA and } N \text{ accepts } w \}$
$M_{\text{dfa}}^{\emptyset}$	$E_{\text{DFA}} = \{ \langle D \rangle : D \text{ is a DFA and } L(D) = \emptyset \}$
$M_{\text{dfa}}^{\bar{=}}$	$EQ_{\text{DFA}} = \{ \langle D_1, D_2 \rangle : D_1, D_2 \text{ are DFAs and } L(D_1) = L(D_2) \}$
M_{cfg}	$A_{\text{CFG}} = \{ \langle G, w \rangle : G \text{ is a CFG and } w \in L(G) \}$
$M_{\text{cfg}}^{\emptyset}$	$E_{\text{CFG}} = \{ \langle G \rangle : G \text{ is a CFG and } L(G) = \emptyset \}$

Figure 2: Language-deciding TMs to know.