Closure Properties

Regular (DFA, NFA, RegEx)

- ➔ Closed: $\cup$ $\cap$ $\circ$ $*$ $\overline{comp}$ $reverse$

Context Free (CFG)

- ➔ Closed: $\cup$ $\circ$ $*$ $reverse$
- ➔ Not closed: $\cap$ $\overline{comp}$

Decidable

- ➔ Closed: $\cup$ $\cap$ $\circ$ $*$ $\overline{comp}$

Recognizable

- ➔ Closed: $\cup$ $\cap$ $\circ$ $*$
- ➔ Not closed: $\overline{comp}$

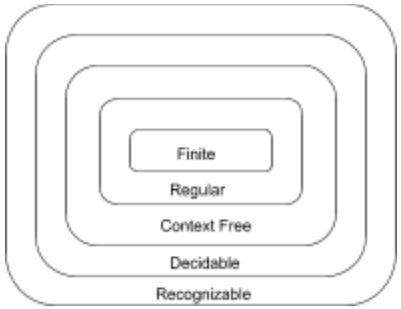
Ex. True: If A is regular and B is regular, A$\cup$B is regular.

Ex. False: If A is regular and A$\cup$B is regular, B is regular

- ➔ Counter example: B is any non-regular language over Σ and Σ^* is regular, Σ^* $\cup$B is regular.

Theorem: If A is a CFL and B is regular, then A$\cap$B is a CFL.

Theorem: If L is recognizable and $\overline{L}$ is recognizable, then L is decidable.



Some True and False

If A is regular and A$\cap$B is regular, then B is regular. False.

If A is regular and A$\cup$B is finite, then B is regular. True.

If A is regular and A$\cap$B is finite, then B is regular. False.

If A is regular and B-A is finite, then B is regular. False.

If A is regular and B is decidable, then A-B is recognizable. True.

If A is regular and B is context free, then A$\cap$B is context free. True.

Reduction

A reduces to B if there exists some function $f : \Sigma \rightarrow \Sigma^*$ such that

$$w \in A \Leftrightarrow f(w) \in B$$

Reducing from $Halt$ to some undecidable language:

If A is $Halt$ and f can be defined, then we can show that f is a reduction from Halt to some language B.

Given any input $< M, w >$ to Halt, we can construct a machine

$$M' = f(< M, w >).$$

If you can construct the reduction f, and you determine if

$< M, w > \in Halt$ then $< M' > \in B$ and if

$< M, w > \notin Halt$ then $< M' > \notin B$,

then this shows that $< M' >= f(< M, w >)$ is a reduction from $Halt$ to B, therefore B is undecidable.

Ex. See last page of the exam.

Note: For problems of the form $\{< M > | (condition)\}$ where the $(condition)$ is to accept "something" and *NOT* with the keywords "at most" or "exactly" some quantity, we can construct $M' = f(< M, w >)$ as follows:

For any input y:

1. Erase the input.
2. Run M on w.
3. Accept if M halts on w.

Ex. $RevAccept$ = $\{< M > | M \text{ accepts } < M >^R\}$

Solution: Assume $RevAccept$ is decidable. We show that this implies that we can decide $Halt$. Given any input $< M, w >$ to $Halt$, we construct the machine $M' = f(< M, w >)$ that, on any input y:

1. Erase input y.
2. Run M on w.
3. M' accepts if M halts on w.

If M halts on w, then $L(M') = \{all\ strings\ y\}$, so $< M' > \in RevAccept$.

Conversely, if M does not halt on w, then $L(M') = \{no\ strings\}$, so $< M' > \notin RevAccept$. This shows that $< M' >= f(< M, w >)$ is a reduction from $Halt$ to $RevAccept$. Since $Halt$ is undecidable, we conclude that $RevAccept$ is undecidable.

Note: For questions that ask for exactly a certain number of strings, you can suggest a machine M' that accepts that many strings.

Ex. $L = \{< M > | M \text{ accepts exactly } 3 \text{ strings}\}$

Partial solution: You can say that $L\{M'\} = \{\varepsilon, 0, 1\}$

Then you construct the reduction as follows (assuming you have written the preamble for Halt, etc.):

For any input y:

1. Run M on w.
2. If M halts on input w, run M' on y.
    a. If y = $\{\varepsilon, 0, 1\}$ then accept
    b. Otherwise, reject.

DFA Transformations

Tip: When describing a transformation of a regular language, we always use the old DFA and describe transformations from the old DFA to solve for a new DFA or NFA.

Ex.

$B \leftarrow^1 C = \{w \in B \mid for\ some\ y \in C,\ strings\ w\ and\ y\ have\ an\ equal\ number\ of\ 1's\}$

Note: Do not let the name $B \leftarrow^1 C$ scare you; it is just a name for the language. $B \leftarrow^1 C$ could have easily been called L or some other name.

To further explain the question,

If $B = \{1011, 00, 01\}$ and

$C = \{00111, \varepsilon, 11\}$,

then $B \leftarrow^1 C = \{1011, 00\}$.

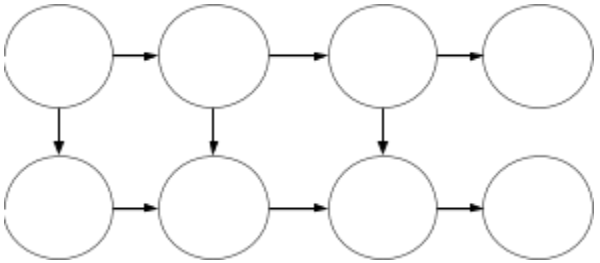
Solution:

1. $Q = Q_B \times Q_C$
2. For $(q, r) \in Q$ and $a \in \Sigma$, where q and r are the current states in B and C respectively, we define:

$$\delta((q, r), a) = \{$$
$$(\delta_B(q, 0), r) \rightarrow if\ a = 0$$
$$(\delta_B(q, 1), \delta_C(r, 1)) \rightarrow if\ a = 1$$
$$(q, \delta_C(r, 0)) \rightarrow if\ a = \varepsilon$$
$$\}$$

3. $q_o = (q_{oB}, q_{oC})$
4. $q_F = (q_{FB}, q_{FC})$

Note: For questions that ask for a DFA with specific quantities of 2 different symbols, do similar to the following:

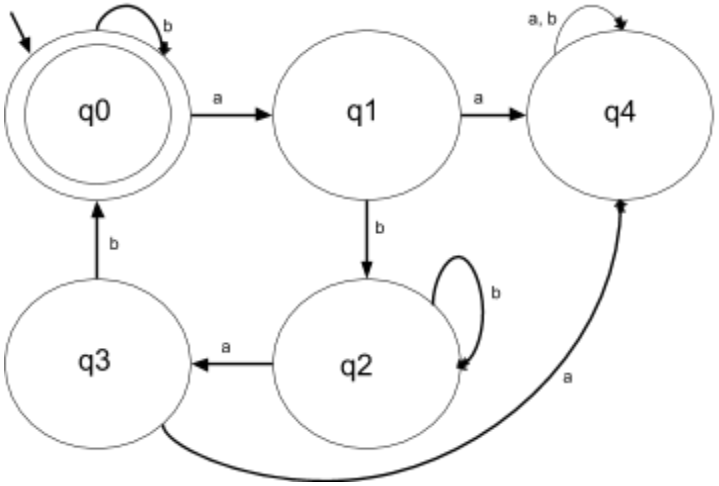


It won't be exactly as above. However, the idea is that you will likely need multiple paths and/or a trap state.

Ex. Draw a DFA for

$L = \{w \mid w\ has\ an\ even\ number\ of\ a's\ and\ each\ a\ is\ followed\ by\ at\ least\ one\ b\}$

Solution:



Decidability

Tips:

1. If we can simulate an arbitrary TM while avoiding the target behaviour, then the behavior is undecidable.
2. If a TM with the target behavior is limited to a finite number of configurations, then the behavior is likely decidable.

Ex.

$$S = \{$$
$$0\ if\ life\ never\ will\ be\ found\ on\ Mars$$
$$1\ if\ life\ will\ be\ found\ on\ Mars\ someday$$
$$\}$$

Is $A = \{S\}$ decidable? Yes it is decidable because $A = \{0\}$ or $A = \{1\}$. It does not matter if we don't know whether 0 or 1 will be chosen, just that 0 or 1 are the only two decisions that can be made.
Ex. $Halt$ is recognizable, but not decidable.
Since $Halt$ is undecidable, we can reduce from $Halt$ to some other language L to show that L is undecidable.

Other examples that are decidable/undecidable:
Determining whether a two-tape TM ever writes a non-blank symbol (or a blank symbol over a non-blank symbol) on its second tape when run on some input w is undecidable.
Determining whether a TM on input w ever moves its head left when its head is on the leftmost tape cell is undecidable.
Determining whether a TM on input w ever moves its head left at any point during its computation on input w is decidable.
Let A, B be two languages such that A ∩ B = ∅ and A and B are recognizable. There exists a decidable language C such that A ⊆ C and B ⊆ C.
Let A = {(n, m)| Every n-state machine M either halts in less than m steps on an empty input, or it does not halt on an empty input }. The language A is undecidable.

Recognizability vs. Decidability
A language is recognizable iff there is a Turing Machine which will halt and accept *only* the strings in that language. For strings not in the language the TM either rejects or runs forever.
From the prof, *"Something is recognizable if there is a way to try all possible inputs in parallel, and eventually you will have the behavior and accept."*
A language is decidable iff there is a Turing Machine which will accept strings in the language and reject strings not in the language.
A language L is decidable iff both L and $\overline{L}$ are recognizable.
Ex. $\overline{Halt}$ is not recognizable.
Since $\overline{Halt}$ is unrecognizable, we can reduce from $\overline{Halt}$ to some other language L to show that L is unrecognizable. Use the above tips for reduction.

Tips for creating CFGs

1. Understand the language, try to describe it in the simplest way possible (my best strategy).
2. If it helps (and you're really good at DFAs or NFAs) draw a DFA or an NFA and convert that to a CFG.

   Ex. Using the DFA from an above example, the CFG for that language is as follows:

$$S \rightarrow bS \mid A \mid \varepsilon$$
$$A \rightarrow abBabBS$$
$$B \rightarrow bB \mid \varepsilon$$

3. Use the following rules for constructing a CFG from a RE:
   a. Single terminal: If RE is a, then the CFG will be $S \rightarrow a$
   b. Union: If RE contains $a + b$, then the CFG will have $S \rightarrow a \mid b$
   c. Concatenation: If RE contains ab, then the CFG will have $S \rightarrow ab$
   d. Star $*$: If RE contains a^*, then the CFG will have $S \rightarrow aS \mid \varepsilon$
   e. Plus+: If RE contains $a+$, then the CFG will have $S \rightarrow aS \mid a$
   f. Star on union: If RE contains $(a+b)^*$, then the CFG will have $S \rightarrow aS \mid bS \mid \varepsilon$
   g. Plus on union: If RE contains $(a+b)^+$, then the CFG will have $S \rightarrow aS \mid bS \mid a \mid b$
   h. Star on concatenation: If RE contains $(ab)^*$, then the CFG will have $S \rightarrow abS \mid \varepsilon$
   i. Plus on concatenation: If RE contains $(ab)^+$, then the CFG will have $S \rightarrow abS \mid ab$

   Of course, these terminals will have different letters and you will have to connect the terminals yourself.
   Ex. Design a CFG for $L = \{a^i b^j c^k \mid i + k < j\}$.
   To describe this language in plain english, we can say that the language L consists of all strings of the form $a^i b^j c^k$ where the number of b's in a string is more than half the length of that string, OR even more simply, for every a or c there is a b, and there is at least one b without a "paired" a or c.
   Solution:

$$S \rightarrow ABC$$
$$A \rightarrow aAb \mid \varepsilon$$
$$B \rightarrow bB \mid b$$
$$C \rightarrow bCc \mid \varepsilon$$