Lecture 18

Further discussion of computability

In this lecture we will discuss a few aspects of decidable and semidecidable languages that were not mentioned in previous lectures. In particular, we will discuss closure properties of these classes of languages and prove a useful alternative characterization of semidecidable languages.

18.1 Closure properties of decidable and semidecidable languages

The decidable and semidecidable languages are closed under many of the operations on languages that we’ve considered thus far in the course (although not all). While we won’t go through every operation we’ve discussed, it is worthwhile to mention some basic examples.

Closure properties of decidable languages

First let us observe that the decidable languages are closed under the regular operations as well as complementation. In short, if $A$ and $B$ are decidable, then there is no difficulty in deciding the languages $A \cup B$, $AB$, $A^*$, and $\overline{A}$ in a straightforward way.

Proposition 18.1. Let $\Sigma$ be an alphabet and let $A, B \subseteq \Sigma^*$ be decidable languages. The languages $A \cup B$, $AB$, and $A^*$ are decidable.

Proof. Because the languages $A$ and $B$ are decidable, there must exist a DSM $M_A$ that decides $A$ and a DSM $M_B$ that decides $B$. The DSMs described in Figures 18.1, 18.2, and 18.3 decide the languages $A \cup B$, $AB$, and $A^*$, respectively. It follows that these languages are all decidable. \qed
The DSM $M$ operates as follows on input $w \in \Sigma^*$:

1. Run $M_A$ on input $w$. If $M_A$ accepts, then accept.
2. Run $M_B$ on input $w$. If $M_B$ accepts, then accept.
3. Reject.

Figure 18.1: A DSM $M$ that decides $A \cup B$, given DSMs $M_A$ and $M_B$ that decide $A$ and $B$, respectively.

The DSM $M$ operates as follows on input $w \in \Sigma^*$:

1. For every choice of strings $u, v \in \Sigma^*$ satisfying $w = uv$:
   1.1 Run $M_A$ on input $u$ and run $M_B$ on input $v$.
   1.2 If both $M_A$ and $M_B$ accept, then accept.
2. Reject.

Figure 18.2: A DSM $M$ that decides $AB$, given DSMs $M_A$ and $M_B$ that decide $A$ and $B$, respectively.

The DSM $M$ operates as follows on input $w \in \Sigma^*$:

1. If $w = \epsilon$, then accept.
2. For every way of writing $w = u_1 \cdots u_m$ for nonempty strings $u_1, \ldots, u_m$:
   2.1 Run $M_A$ on each of the strings $u_1, \ldots, u_m$.
   2.2 If $M_A$ accepts all of the strings $u_1, \ldots, u_m$, then accept.
3. Reject.

Figure 18.3: A DSM $M$ that decides $A^*$, given a DSM $M_A$ that decides $A$. 

2
The decidable languages are also closed under complementation, as the next proposition states. Perhaps we don’t even need to bother writing a proof for this one: if a DSM $M$ decides $A$, then one can obtain a new DSM $K$ deciding $\overline{A}$ by defining $K$ so that it runs $M$ on a given string and accepts if and only if $M$ rejects.

**Proposition 18.2.** Let $\Sigma$ be an alphabet and let $A \subseteq \Sigma^*$ be a decidable language. The language $\overline{A}$ is decidable.

There are a variety of other operations under which the decidable languages are closed. For example, because the decidable languages are closed under union and complementation, we immediately have that they are closed under intersection and symmetric difference. Another example is string reversal: if a language $A$ is decidable, then $A^R$ is also decidable, because a DSM can decide $A^R$ simply by reversing the input string, then deciding whether the string that is obtained is contained in $A$.

There are, however, some natural operations under which the decidable languages are not closed. The following example shows that this is the case for the prefix operation.

**Example 18.3.** The language Prefix($A$) might not be decidable, even if $A$ is decidable. To construct an example that illustrates that this is so, let us first take $B \subseteq \{0, 1\}^*$ to be any language that is semidecidable but not decidable (such as HALT).

Let $M_B$ be a DSM such that $L(M_B) = B$, and define a language $A \subseteq \{0, 1, \#\}^*$ as follows:

$$A = \{w\#0^t : M_B \text{ accepts } w \text{ within } t \text{ steps} \}. \quad (18.1)$$

This is a decidable language, but Prefix($A$) is not—for if Prefix($A$) were decidable, then one could easily decide $B$ by using the fact that a string $w \in \{0, 1\}^*$ is contained in $B$ if and only if $w\# \in$ Prefix($A$). (That is, $w \in B$ and $w\# \in$ Prefix($A$) are both equivalent to the existence of a positive integer $t$ such that $w\#0^t \in A$.)

**Closure properties of semidecidable languages**

The semidecidable languages are also closed under a variety of operations, although not precisely the same operations under which the decidable languages are closed.

Let us begin with the regular operations, under which the semidecidable languages are indeed closed. In this case, one needs to be a bit more careful than was sufficient when proving the analogous property for decidable languages, as the stack machines that recognize these languages might run forever.
The DSM \( M \) operates as follows on input \( w \in \Sigma^* \):

1. Set \( t \leftarrow 1 \).

2. Run \( M_A \) on input \( w \) for \( t \) steps. If \( M_A \) accepts \( w \) within \( t \) steps, then accept.

3. Run \( M_B \) on input \( w \) for \( t \) steps. If \( M_B \) accepts \( w \) within \( t \) steps, then accept.

4. Set \( t \leftarrow t + 1 \) and goto 2.

Figure 18.4: A DSM \( M \) that semidecides \( A \cup B \), given DSMs \( M_A \) and \( M_B \) that semidecide \( A \) and \( B \), respectively.

The DSM \( M \) operates as follows on input \( w \in \Sigma^* \):

1. Set \( t \leftarrow 1 \).

2. For every choice of strings \( u, v \) satisfying \( w = uv \):
   1.1 Run \( M_A \) on input \( u \) for \( t \) steps and run \( M_B \) on input \( v \) for \( t \) steps.
   1.2 If both \( M_A \) and \( M_B \) have accepted within \( t \) steps, then accept.

2. Set \( t \leftarrow t + 1 \) and goto 2.

Figure 18.5: A DSM \( M \) that semidecides \( AB \), given DSMs \( M_A \) and \( M_B \) that semidecide \( A \) and \( B \), respectively.

**Proposition 18.4.** Let \( \Sigma \) be an alphabet and let \( A, B \subseteq \Sigma^* \) be semidecidable languages. The languages \( A \cup B, AB, \) and \( A^* \) are semidecidable.

**Proof.** Because the languages \( A \) and \( B \) are semidecidable, there must exist DSMs \( M_A \) and \( M_B \) such that \( L(M_A) = A \) and \( L(M_B) = B \). The DSMs described in Figures 18.4, 18.5, and 18.6 semidecide the languages \( A \cup B, AB, \) and \( A^* \), respectively. It follows that these languages are all semidecidable.

The semidecidable languages are also closed under intersection. This can be proved through a similar method to closure under union, but in fact this is a situation in which we don’t actually need to be as careful about running forever.
The DSM \( M \) operates as follows on input \( w \in \Sigma^* \):

1. If \( w = \epsilon \), then accept.
2. Set \( t \leftarrow 1 \).
3. For every way of writing \( w = u_1 \cdots u_m \) for nonempty strings \( u_1, \ldots, u_m \):
   
   3.1 Run \( M_A \) on each of the strings \( u_1, \ldots, u_m \) for \( t \) steps.
   
   3.2 If \( M_A \) accepts all of the strings \( u_1, \ldots, u_m \) within \( t \) steps, then accept.
4. Set \( t \leftarrow t + 1 \) and goto 3.

**Figure 18.6**: A DSM \( M \) that semidecides \( A^* \), given a DSM \( M_A \) that semidecides \( A \).

The DSM \( M \) operates as follows on input \( w \in \Sigma^* \):

1. Run \( M_A \) on input \( w \). If \( M_A \) rejects \( w \), then reject.
2. Run \( M_B \) on input \( w \). If \( M_B \) rejects \( w \), then reject.
3. Accept.

**Figure 18.7**: A DSM \( M \) that semidecides \( A \cap B \), given DSMs \( M_A \) and \( M_B \) that semidecide \( A \) and \( B \), respectively. Note that if either \( M_A \) or \( M_B \) runs forever on input \( w \), then so does \( M \), but this does not change the fact that \( M \) semidecides \( A \cap B \).

**Proposition 18.5.** Let \( \Sigma \) be an alphabet and let \( A, B \subseteq \Sigma^* \) be semidecidable languages. The language \( A \cap B \) is semidecidable.

**Proof.** Because the languages \( A \) and \( B \) are semidecidable, there must exist DSMs \( M_A \) and \( M_B \) such that \( L(M_A) = A \) and \( L(M_B) = B \). The DSM \( M \) described in Figure 18.7 semidecides \( A \cap B \), which implies that \( A \cap B \) is semidecidable. \( \Box \)

It turns out that the semidecidable languages are not closed under complementation. We will be able to conclude this from the following theorem, which is both interesting in its own right and useful in other situations.
The DSM $M$ operates as follows on input $w \in \Sigma^*$:

1. Set $t \leftarrow 1$.
2. Run $M_0$ for $t$ steps on input $w$. If $M_0$ accepts within $t$ steps, then accept.
3. Run $M_1$ for $t$ steps on input $w$. If $M_1$ accepts within $t$ steps, then reject.
4. Set $t \leftarrow t + 1$ and goto 2.

Figure 18.8: A DSM $M$ that decides $A$, given DSMs $M_0$ and $M_1$ that semidecide $A$ and $\overline{A}$, respectively.

**Theorem 18.6.** Let $\Sigma$ be an alphabet and let $A \subseteq \Sigma^*$ be a language such that both $A$ and $\overline{A}$ are semidecidable. The language $A$ is decidable.

*Proof.* Because $A$ and $\overline{A}$ are semidecidable languages, there must exist DSMs $M_0$ and $M_1$ such that $A = L(M_0)$ and $\overline{A} = L(M_1)$. Define a new DSM $M$ as described in Figure 18.8.

Now let us consider the behavior of the DSM $M$ on a given input string $w$. If it is the case that $w \in A$, then $M_0$ eventually accepts $w$, while $M_1$ does not. (It could be that $M_1$ either rejects or runs forever, but it cannot accept $w$.) It is therefore the case that $M$ accepts $w$. On the other hand, if $w \notin A$, then $M_1$ eventually accepts $w$ while $M_0$ does not, and therefore $M$ rejects $w$. Consequently, $M$ decides $A$, so $A$ is decidable. \qed

We know that there exist languages, such as HALT, that are semidecidable but not decidable—so it cannot be that the semidecidable languages are closed under complementation.

Finally, there are some operations under which the semidecidable languages are closed, but under which the decidable languages are not. For example, if $A$ is semidecidable, then so are the languages Prefix$(A)$, Suffix$(A)$, and Substring$(A)$.

### 18.2 The range of a computable function

We will now consider an alternative characterization of semidecidable languages (with the exception of the empty language), which is that they are precisely the languages that are equal to the range of a computable function. Recall that the range of a function $f : \Gamma^* \rightarrow \Sigma^*$ is defined as follows:

$$\text{range}(f) = \{ f(w) : w \in \Gamma^* \}. \quad (18.2)$$
The DSM $M$ operates as follows on input $w \in \Sigma^*$:

1. Set $x \leftarrow \varepsilon$.
2. Compute $y = f(x)$, and accept if $w = y$.
3. Increment $x$ with respect to the lexicographic ordering of $\Gamma^*$ and goto 2.

Figure 18.9: A DSM $M$ that semidecides $A = \text{range}(f)$ for a computable function $f: \Gamma^* \rightarrow \Sigma^*$.

**Theorem 18.7.** Let $\Sigma$ and $\Gamma$ be alphabets and let $A \subseteq \Sigma^*$ be a nonempty language. The following two statements are equivalent:

1. $A$ is semidecidable.
2. There exists an a computable function $f: \Gamma^* \rightarrow \Sigma^*$ such that $A = \text{range}(f)$.

**Proof.** Let us first prove that the second statement implies the first. That is, we will prove that if there exists a computable function $f: \Gamma^* \rightarrow \Sigma^*$ such that $A = \text{range}(f)$, then $A$ is semidecidable. Consider the DSM $M$ described in Figure 18.9. In essence, this DSM searches over $\Gamma^*$ to find a string that $f$ maps to a given input string $w$. If it is the case that $w \in \text{range}(f)$, then $M$ will eventually find $x \in \Gamma^*$ such that $f(x) = w$ and accept, while $M$ will certainly not accept if $w \notin \text{range}(f)$. Thus, we have $L(M) = \text{range}(f) = A$, which implies that $A$ is semidecidable.

Now suppose that $A$ is semidecidable, so that there exists a DSM $M$ such that $L(M) = A$. We will also make use of the assumption that $A$ is nonempty—there exists at least one string in $A$, so we may take $w_0$ to be such a string. (If you like, you may define $w_0$ more concretely as the first string in $A$ with respect to the lexicographic ordering of $\Sigma^*$, but it is not important for the proof that we make this particular choice.) Define a function $f: \Gamma^* \rightarrow \Sigma^*$ as follows:

$$f(x) = \begin{cases} 
  w & \text{if } x = \langle w, 0^t \rangle, \text{ for } w \in \Sigma^* \text{ and } t \in \mathbb{N}, \text{ and } M \text{ accepts } w \text{ within } t \text{ steps} \\
  w_0 & \text{otherwise.}
\end{cases}$$

(18.3)

Here we assume that $\langle w, 0^t \rangle$ refers to any encoding scheme through which the strings $w \in \Sigma^*$ and $0^t \in \{0\}^*$ may be encoded into a single string $\langle w, 0^t \rangle \in \Gamma^*$. (As we discussed earlier in the course, this is possible even if $\Gamma$ contains only a single symbol.) It is evident that the function $f$ is computable: a DSM $M_f$ can compute $f$ by checking to see if the input has the form $\langle w, 0^t \rangle$, simulating $M$ for $t$ steps on
input \( w \) if so, and then outputting either \( w \) or \( w_0 \) depending on the outcome. If \( M \) accepts a particular string \( w \), then it must be that \( w = f(\langle w, 0^t \rangle) \) for some sufficiently large natural number \( t \), so \( A \subseteq \text{range}(f) \). On the other hand, every output of \( f \) is either a string \( w \) accepted by \( M \) or the string \( w_0 \), and therefore \( \text{range}(f) \subseteq A \). It therefore holds that \( A = \text{range}(f) \), which completes the proof.

**Remark 18.8.** The assumption that \( A \) is nonempty is essential in the previous theorem because it cannot be that \( \text{range}(f) = \emptyset \) for a computable function \( f \). Indeed, it cannot be that \( \text{range}(f) = \emptyset \) for any function whatsoever.

Theorem 18.7 provides a useful characterization of semidecidable languages. For instance, you can use this theorem to come up with alternative proofs for all of the closure properties of the semidecidable languages stated in the previous section.

For example, suppose that \( A, B \subseteq \Sigma^* \) are nonempty semidecidable languages. By the theorem above, there must exist computable functions \( f : \Sigma^* \to \Sigma^* \) and \( g : \Sigma^* \to \Sigma^* \) such that \( \text{range}(f) = A \) and \( \text{range}(g) = B \). Define a new function \( h : (\Sigma \cup \{\#\})^* \to \Sigma^* \) as follows:

\[
g(x) = \begin{cases} f(y)f(z) & \text{if } x = y#z \text{ for } y, z \in \Sigma^* \\ f(\epsilon)g(\epsilon) & \text{otherwise.} \end{cases}
\] (18.4)

(We are assuming \( # \notin \Sigma \), but if it were, \# could of course be replaced by any other choice of a symbol that is not contained in \( \Sigma \).) One sees that \( h \) is computable, and \( \text{range}(h) = AB \), which implies that \( AB \) is semidecidable.

Here is another example of an application of Theorem 18.7.

**Corollary 18.9.** Let \( \Sigma \) be an alphabet and let \( A \subseteq \Sigma^* \) be any infinite semidecidable language. There exists an infinite decidable language \( B \subseteq A \).

**Proof.** Because \( A \) is infinite (and therefore nonempty), there exists a computable function \( f : \Sigma^* \to \Sigma^* \) such that \( A = \text{range}(f) \).

We will define a language \( B \) by first defining a DSM \( M \) and then taking \( B = L(M) \). In order for us to be sure that \( B \) satisfies the requirements of the corollary, it will need to be proved that \( M \) never runs forever (so that \( B \) is decidable), that \( M \) only accepts strings that are contained in \( A \) (so that \( B \subseteq A \)), and that \( M \) accepts infinitely many different strings (so that \( B \) is infinite). The DSM \( M \) is described in Figure 18.10.

The fact that \( M \) never runs forever follows from the assumption that \( A \) is infinite. That is, because \( A \) is infinite, the function \( f \) must output infinitely many different strings, so regardless of what input string \( w \) is input into \( M \), the loop will
Lecture 18

The DSM $M$ operates as follows on input $w \in \Sigma^*$:

1. Set $x \leftarrow \varepsilon$.
2. Compute $y \leftarrow f(x)$.
3. If $y = w$ then accept.
4. If $y > w$ (with respect to the lexicographic ordering of $\Sigma^*$) then reject.
5. Increment $x$ with respect to the lexicographic ordering of $\Sigma^*$ and goto 2.

Figure 18.10: A DSM $M$ for Corollary 18.9.

eventually reach a string $x$ so that $f(x) = w$ or $f(x) > w$, either of which causes $M$ to halt.

The fact that $M$ only accepts strings in $A$ follows from the fact that the condition for acceptance is that the input string $w$ is equal to $y$, which is contained in $\text{range}(f) = A$.

Finally, let us observe that $M$ accepts precisely the strings in this set:

$$\left\{ w \in \Sigma^* : \text{there exists } x \in \Sigma^* \text{ such that } w = f(x) \right\}.$$  \hspace{1cm} (18.5)

The fact that this set is infinite follows from the assumption that $A = \text{range}(f)$ is infinite—for if the set were finite, there would necessarily be a maximal output of $f$ with respect to the lexicographic ordering of $\Sigma^*$, contradicting the assumption that $\text{range}(f)$ is infinite.

The language $B = L(M)$ therefore satisfies the requirements of the corollary, which completes the proof. \hfill \square