

## Section 5.2: Closure Properties of Recursive and Recursively Enumerable Languages

In this section, we will see that the recursive and recursively enumerable languages are closed under union, concatenation, closure and intersection. The recursive languages are also closed under set difference and complementation. In the next section, we will see that the recursively enumerable languages are not closed under complementation or set difference. On the other hand, we will see in this section that, if a language and its complement are both r.e., then the language is recursive.

---

Copyright © 2003–4 Alley Stoughton

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

The  $\text{\LaTeX}$  source of these slides, the associated book, and the distribution of the Forlan toolset are available on the WWW at <http://people.cis.ksu.edu/~stough/forlan/>.

## (5.2) Closure Properties of Recursive Languages

### Theorem 5.2.1

If  $L$ ,  $L_1$  and  $L_2$  are recursive languages, then so are  $L_1 \cup L_2$ ,  $L_1L_2$ ,  $L^*$ ,  $L_1 \cap L_2$  and  $L_1 - L_2$ .

**Proof.** Let's consider the concatenation case as an example. Since  $L_1$  and  $L_2$  are recursive languages, there are programs  $P_1$  and  $P_2$  that test whether strings are in  $L_1$  and  $L_2$ , respectively. We combine the functions of  $P_1$  and  $P_2$  to form a program  $Q$  that takes in a string  $w$  and behaves as follows. First, it generates all of the pairs of strings  $(x, y)$  such that  $xy = w$ . Then it works through these pairs, one by one. Given such a pair  $(x, y)$ ,  $Q$  calls the principal function of  $P_1$  to check whether  $x \in L_1$ . If the answer is "no", then it goes on to the next pair. Otherwise, it calls the principal function of  $P_2$  to check whether  $y \in L_2$ . If the answer is "no", then it goes on to the next pair. Otherwise, it returns **true**. If  $Q$  runs out of pairs to check, then it returns **false**.

## (5.2) Closure Properties of Recursive Languages (Cont.)

**Proof (cont.).** We can check that, for all  $w \in \mathbf{Str}$ ,  $Q$  tests whether  $w \in L_1L_2$ . Thus  $L_1L_2$  is recursive.  $\square$

### Corollary 5.2.2

*If  $\Sigma$  is an alphabet and  $L \subseteq \Sigma^*$  is recursive, then so is  $\Sigma^* - L$ .*

**Proof.** Follows from Theorem 5.2.1, since  $\Sigma^*$  is recursive.  $\square$

## (5.2) Closure Properties of Recursively Enumerable Languages

### Theorem 5.2.3

If  $L$ ,  $L_1$  and  $L_2$  are recursively enumerable languages, then so are  $L_1 \cup L_2$ ,  $L_1L_2$ ,  $L^*$  and  $L_1 \cap L_2$ .

**Proof.** We consider the concatenation case as an example. Since  $L_1$  and  $L_2$  are recursively enumerable, there are programs  $P_1$  and  $P_2$  such that, for all  $w \in \mathbf{Str}$ ,  $w \in L_1$  iff  $\mathbf{run}_1(P_1, w) = \mathbf{true}$ , and  $w \in L_2$  iff  $\mathbf{run}_1(P_2, w) = \mathbf{true}$ . (Remember that  $P_1$  and  $P_2$  may fail to terminate on some inputs.) To show that  $L_1L_2$  is recursively enumerable, we will construct a program  $Q$  such that, for all  $w \in \mathbf{Str}$ ,  $w \in L_1L_2$  iff  $\mathbf{run}_1(Q, w) = \mathbf{true}$ .

## (5.2) Closure Properties of Recursively Enumerable Languages (Cont.)

**Proof (cont.).** When  $Q$  is called with a string  $w$ , it behaves as follows. First, it generates all the pairs of strings  $(x, y)$  such that  $w = xy$ . Let these pairs be  $(x_1, y_1), \dots, (x_n, y_n)$ .

## (5.2) Closure Properties of Recursively Enumerable Languages (Cont.)

**Proof (cont.).** When  $Q$  is called with a string  $w$ , it behaves as follows. First, it generates all the pairs of strings  $(x, y)$  such that  $w = xy$ . Let these pairs be  $(x_1, y_1), \dots, (x_n, y_n)$ . Now,  $Q$  uses our *incremental* interpretation function to work its way through all the string pairs, running  $P_1$  with input  $x_1$  (for some fixed number of steps),  $P_2$  with input  $y_1$ ,  $P_1$  with input  $x_2$ ,  $P_2$  with input  $y_2$ ,  $\dots$ ,  $P_1$  with input  $x_n$  and  $P_2$  with input  $y_n$ . It then begins a second round of all of these incremental interpretations, followed by a third round, and so on.

- If, at some stage, the incremental interpretation of  $P_1$  on input  $x_i$  returns **true**, then  $x_i$  is marked as being in  $L_1$ .
- If, at some stage, the incremental interpretation of  $P_2$  on input  $y_i$  returns **true**, then the  $y_i$  is marked as being in  $L_2$ .

## (5.2) Closure Properties of Recursively Enumerable Languages (Cont.)

### Proof (cont.).

- If, at some stage, the incremental interpretation of  $P_1$  on input  $x_i$  terminates with **false** or **error**, then the  $i$ 'th pair is marked as discarded.
- If, at some stage, the incremental interpretation of  $P_2$  on input  $y_i$  terminates with **false** or **error**, then the  $i$ 'th pair is marked as discarded.
- If, at some stage,  $x_i$  is marked as in  $L_1$  and  $y_i$  is marked as in  $L_2$ , then  $Q$  returns **true**.
- If, at some stage, there are no remaining pairs, then  $Q$  returns **false**.

We can check that for all  $w \in \mathbf{Str}$ ,  $w \in L_1L_2$  iff  $\mathbf{run}_1(Q, w) = \mathbf{true}$ .

□

## (5.2) Closure Properties of Recursively Enumerable Languages (Cont.)

### Theorem 5.2.4

If  $\Sigma$  is an alphabet,  $L \subseteq \Sigma^*$  is a recursively enumerable language, and  $\Sigma^* - L$  is recursively enumerable, then  $L$  is recursive.

**Proof.** Since  $L$  and  $\Sigma^* - L$  are recursively enumerable languages, there are programs  $P$  and  $P'$  such that, for all  $w \in \mathbf{Str}$ ,  $w \in L$  iff  $\mathbf{run}_1(P, w) = \mathbf{true}$ , and  $w \in \Sigma^* - L$  iff  $\mathbf{run}_1(P', w) = \mathbf{true}$ . We construct a program  $Q$  that behaves as follows when called with a string  $w$ . If  $w \notin \Sigma^*$ , then  $Q$  returns **false**. Otherwise,

## (5.2) Closure Properties of Recursively Enumerable Languages (Cont.)

### Theorem 5.2.4

If  $\Sigma$  is an alphabet,  $L \subseteq \Sigma^*$  is a recursively enumerable language, and  $\Sigma^* - L$  is recursively enumerable, then  $L$  is recursive.

**Proof.** Since  $L$  and  $\Sigma^* - L$  are recursively enumerable languages, there are programs  $P$  and  $P'$  such that, for all  $w \in \mathbf{Str}$ ,  $w \in L$  iff  $\mathbf{run}_1(P, w) = \mathbf{true}$ , and  $w \in \Sigma^* - L$  iff  $\mathbf{run}_1(P', w) = \mathbf{true}$ . We construct a program  $Q$  that behaves as follows when called with a string  $w$ . If  $w \notin \Sigma^*$ , then  $Q$  returns **false**. Otherwise,  $Q$  alternates between incrementally interpreting  $P$  with input  $w$  and incrementally interpreting  $P'$  with input  $w$ .

## (5.2) Closure Properties of Recursively Enumerable Languages (Cont.)

### Proof (cont.).

- If, at some stage, the incremental interpretation of  $P$  returns **true**, then  $Q$  returns **true**.
- If, at some stage, the incremental interpretation of  $P$  returns **false** or **error**, then  $Q$  returns **false**.
- If, at some stage, the incremental interpretation of  $P'$  returns **true**, then  $Q$  returns **false**.
- If, at some stage, the incremental interpretation of  $P'$  returns **false** or **error**, then  $Q$  returns **true**.

We can check that, for all  $w \in \mathbf{Str}$ ,  $Q$  tests whether  $w \in L$ .  $\square$