

Section 4.7: Closure Properties of Context-free Languages

In this section, we define union, concatenation and closure operations on grammars. As a result, we will have that the context-free languages are closed under union, concatenation and closure.

Later, we will see that the context-free languages aren't closed under intersection, complementation, and set difference.

The book shows several additional closure properties of context-free languages, in addition to giving the corresponding operations on grammars.

Copyright © 2003–7 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 \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/>.

(4.7) Basic Grammars and Operations on Grammars

The grammar with variable A and production $A \rightarrow \%$ generates the language $\{\%\}$.

The grammar with variable A and no productions generates the language \emptyset .

If w is a string, then the grammar with variable A and production $A \rightarrow w$ generates the language $\{w\}$. Actually, we must be careful to choose a variable that doesn't occur in w .

(4.7) Union Operation

Suppose G_1 and G_2 are grammars. We can define a grammar H such that $L(H) = L(G_1) \cup L(G_2)$ by unioning together the variables and productions of G_1 and G_2 , and adding a new start variable q , along with productions

(4.7) Union Operation

Suppose G_1 and G_2 are grammars. We can define a grammar H such that $L(H) = L(G_1) \cup L(G_2)$ by unioning together the variables and productions of G_1 and G_2 , and adding a new start variable q , along with productions

$$q \rightarrow s_{G_1} \mid s_{G_2}.$$

What do we have to know about G_1 , G_2 and q for the above to be valid?

(4.7) Union Operation

Suppose G_1 and G_2 are grammars. We can define a grammar H such that $L(H) = L(G_1) \cup L(G_2)$ by unioning together the variables and productions of G_1 and G_2 , and adding a new start variable q , along with productions

$$q \rightarrow s_{G_1} \mid s_{G_2}.$$

What do we have to know about G_1 , G_2 and q for the above to be valid?

- $Q_{G_1} \cap Q_{G_2} = \emptyset$ and $q \notin Q_{G_1} \cup Q_{G_2}$;

(4.7) Union Operation

Suppose G_1 and G_2 are grammars. We can define a grammar H such that $L(H) = L(G_1) \cup L(G_2)$ by unioning together the variables and productions of G_1 and G_2 , and adding a new start variable q , along with productions

$$q \rightarrow s_{G_1} \mid s_{G_2}.$$

What do we have to know about G_1 , G_2 and q for the above to be valid?

- $Q_{G_1} \cap Q_{G_2} = \emptyset$ and $q \notin Q_{G_1} \cup Q_{G_2}$;
- **alphabet** $(G_1) \cap Q_{G_2} = \emptyset$, **alphabet** $(G_2) \cap Q_{G_1} = \emptyset$ and $q \notin$ **alphabet** $(G_1) \cup$ **alphabet** (G_2) .

(4.7) Union Operation (Cont.)

Our official union operation for grammars renames the variables of G_1 and G_2 , and chooses the start variable q , in a uniform way that makes the preceding properties hold.

To keep things simple, when talking about the concatenation and closure operations on grammars, we'll just assume that conflicts between variables and alphabet elements don't occur.

(4.7) Concatenation and Closure Operations

Suppose G_1 and G_2 are grammars. We can define a grammar H such that $L(H) = L(G_1)L(G_2)$ by unioning together the variables and productions of G_1 and G_2 , and adding a new start variable q , along with production

(4.7) Concatenation and Closure Operations

Suppose G_1 and G_2 are grammars. We can define a grammar H such that $L(H) = L(G_1)L(G_2)$ by unioning together the variables and productions of G_1 and G_2 , and adding a new start variable q , along with production

$$q \rightarrow s_{G_1}s_{G_2}.$$

Suppose G is a grammar. We can define a grammar H such that $L(H) = L(G)^*$ by adding to the variables and productions of G a new start variable q , along with productions

(4.7) Concatenation and Closure Operations

Suppose G_1 and G_2 are grammars. We can define a grammar H such that $L(H) = L(G_1)L(G_2)$ by unioning together the variables and productions of G_1 and G_2 , and adding a new start variable q , along with production

$$q \rightarrow s_{G_1} s_{G_2}.$$

Suppose G is a grammar. We can define a grammar H such that $L(H) = L(G)^*$ by adding to the variables and productions of G a new start variable q , along with productions

$$q \rightarrow \% \mid s_G q.$$

(4.7) Summary of Closure Properties

Theorem 4.7.1

Suppose $L, L_1, L_2 \in \mathbf{CFLan}$. Then:

- (1) $L_1 \cup L_2 \in \mathbf{CFLan}$;
- (2) $L_1 L_2 \in \mathbf{CFLan}$;
- (3) $L^* \in \mathbf{CFLan}$.

(4.7) Operations on Grammars in Forlan

The Forlan module `Gram` defines the following constants and operations on grammars:

```
val emptyStr      : gram
val emptySet      : gram
val fromStr       : str -> gram
val fromSym       : sym -> gram
val union         : gram * gram -> gram
val concat        : gram * gram -> gram
val closure       : gram -> gram
```

The functions `fromStr` and `fromSym` are also available in the top-level environment with the names

```
val strToGram : str -> gram
val symToGram : sym -> gram
```

More operations on grammars are described by the book.

(4.7) Forlan Examples

For example, we can construct a grammar G such that $L(G) = \{01\} \cup \{10\}\{11\}^*$, as follows.

```
- val gram1 = strToGram(Str.fromString "01");  
val gram1 = - : gram  
- val gram2 = strToGram(Str.fromString "10");  
val gram2 = - : gram  
- val gram3 = strToGram(Str.fromString "11");  
val gram3 = - : gram  
- val gram =  
= Gram.union(gram1,  
=           Gram.concat(gram2,  
=                   Gram.closure gram3));  
val gram = - : gram
```

(4.7) Forlan Examples (Cont.)

```
- Gram.output("", gram);  
{variables}  
A, <1,A>, <2,A>, <2,<1,A>>, <2,<2,A>>, <2,<2,<A>>>  
{start variable}  
A  
{productions}  
A -> <1,A> | <2,A>; <1,A> -> 01;  
<2,A> -> <2,<1,A>><2,<2,A>>; <2,<1,A>> -> 10;  
<2,<2,A>> -> % | <2,<2,<A>>><2,<2,A>>; <2,<2,<A>>> -> 11  
val it = () : unit
```

(4.7) Forlan Examples (Cont.)

```
- val gram' = Gram.renameVariablesCanonically gram;
val gram' = - : gram
- Gram.output("", gram');
{variables}
A, B, C, D, E, F
{start variable}
A
{productions}
A -> B | C; B -> 01; C -> DE; D -> 10; E -> % | FE;
F -> 11
val it = () : unit
```