CSC 2204 Finite Automata Theory and Formal Languages

Department of Computer Science

SZABIST (Islamabad Campus)

Week 5 (Lecture 2)

Associativity & Commutativity

Commutativity is the property of an operator that says we can switch the order of its operands and get the same result. An example for arithmetic was given above: $x + y = y + x$. *Associativity* is the property of an operator that allows us to regroup the operands when the operator is applied twice. For example, the associative law of multiplication is $(x \times y) \times z = x \times (y \times z)$.

Associativity & Commutativity

$L + M = M + L$. This law, the *commutative law for union*, says that we may take the union of two languages in either order.

$(L + M) + N = L + (M + N)$. This law, the *associative law for union*, says that we may take the union of three languages either by taking the union of the first two initially, or taking the union of the last two initially.

$(LM)N = L(MN)$. This law, the *associative law for concatenation*, says that we can concatenate three languages by concatenating either the first two or the last two initially.

Missing from this list is the "law" $LM = ML$, which would say that concatenation is commutative. However, this law is false.

Identities & Annihilators

An *identity* for an operator is a value such that when the operator is applied to the identity and some other value, the result is the other value. For instance, 0 is the identity for addition, since $0 + x = x + 0 = x$, and 1 is the identity for multiplication, since $1 \times x = x \times 1 = x$. An *annihilator* for an operator is a value such that when the operator is applied to the annihilator and some other value, the result is the annihilator. For instance, 0 is an annihilator for multiplication, since $0 \times x = x \times 0 = 0$. There is no annihilator for addition.

- $\emptyset + L = L + \emptyset = L$. This law asserts that $\emptyset$ is the identity for union.
- $\epsilon L = L\epsilon = L$. This law asserts that ϵ is the identity for concatenation.
- $\emptyset L = L\emptyset = \emptyset$. This law asserts that $\emptyset$ is the annihilator for concatenation.

Powerful Tools for Simplification

Distributive Laws

A *distributive law* involves two operators, and asserts that one operator can be pushed down to be applied to each argument of the other operator individually. The most common example from arithmetic is the distributive law of multiplication over addition, that is, $x \times (y + z) = x \times y + x \times z$.

$L(M + N) = LM + LN$. This law, is the *left distributive law of concatenation over union.*

$(M + N)L = ML + NL$. This law, is the *right distributive law of concatenation over union.*

Distributive Laws

Theorem: If L, M, and N are any languages, then $L(M \cup N) = LM \cup LN$

PROOF: The proof is similar to another proof about a distributive law that we saw in Theorem 1.10. We need first to show that a string w is in $L(M \cup N)$ if and only if it is in $LM \cup LN$.

(Only-if) If w is in $L(M \cup N)$, then $w = xy$, where x is in L and y is in either M or N. If y is in M, then xy is in LM, and therefore in $LM \cup LN$. Likewise, if y is in N, then xy is in LN and therefore in $LM \cup LN$.

(If) Suppose w is in $LM \cup LN$. Then w is in either LM or in LN. Suppose first that w is in LM. Then $w = xy$, where x is in L and y is in M. As y is in M, it is also in $M \cup N$. Thus, xy is in $L(M \cup N)$. If w is not in LM, then it is surely in LN, and a similar argument shows it is in $L(M \cup N)$.

Simplification

Consider the regular expression $\mathbf{0}+\mathbf{01}^*$. We can "factor out a $\mathbf{0}$" from the union, but first we have to recognize that the expression $\mathbf{0}$ by itself is actually the concatenation of $\mathbf{0}$ with something, namely ϵ. That is, we use the identity law for concatenation to replace $\mathbf{0}$ by $\mathbf{0}\epsilon$, giving us the expression $\mathbf{0}\epsilon+\mathbf{01}^*$. Now, we can apply the left distributive law to replace this expression by $\mathbf{0}(\epsilon+\mathbf{1}^*)$. If we further recognize that ϵ is in $L(\mathbf{1}^*)$, then we observe that $\epsilon+\mathbf{1}^*=\mathbf{1}^*$, and can simplify to $\mathbf{01}^*$.

Idempotent Law

An operator is said to be *idempotent* if the result of applying it to two of the same values as arguments is that value. The common arithmetic operators are not idempotent; $x + x \neq x$ in general and $x \times x \neq x$ in general (although there are *some* values of x for which the equality holds, such as $0 + 0 = 0$).

$L + L = L$. This law, the *idempotence law for union*, states that if we take the union of two identical expressions, we can replace them by one copy of the expression.

Laws involving Closure

$(L^*)^* = L^*$. This law says that closing an expression that is already closed does not change the language. The language of $(L^*)^*$ is all strings created by concatenating strings in the language of L^*. But those strings are themselves composed of strings from L. Thus, the string in $(L^*)^*$ is also a concatenation of strings from L and is therefore in the language of L^*.

$\emptyset^* = \epsilon$. The closure of $\emptyset$ contains only the string ϵ

$\epsilon^* = \epsilon$. It is easy to check that the only string that can be formed by concatenating any number of copies of the empty string is the empty string itself.

Laws involving Closure

$L^+ = LL^* = L^*L$. Recall that L^+ is defined to be $L + LL + LLL + \cdots$. Also, $L^* = \epsilon + L + LL + LLL + \cdots$. Thus,

$$LL^* = L\epsilon + LL + LLL + LLLL + \cdots$$

$L^* = L^+ + \epsilon$. The proof is easy, since the expansion of L^+ includes every term in the expansion of L^* except ϵ. Note that if the language L contains the string ϵ, then the additional "$+\epsilon$" term is not needed; that is, $L^+ = L^*$ in this special case.

$L? = \epsilon + L$. This rule is really the definition of the ? operator.