Transfromation of a CFG to Chomsky Normal Form

Jay Bagga

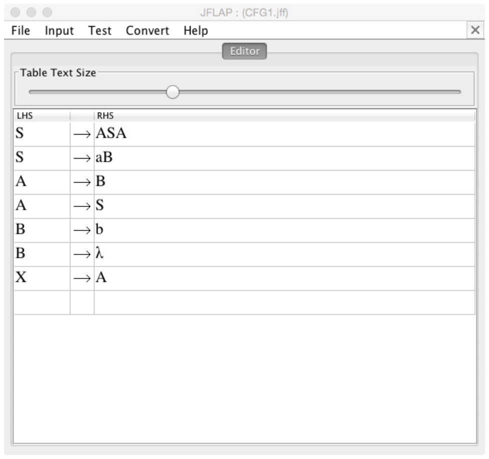
1 Chomsky Normal Form

A CFG is said to be in Chomsky Normal Form (CNF) if every rule is of the form
$A \rightarrow BC$, or
$A \rightarrow a$
where a is a terminal symbol and A, B, C are any variables. In addition, if the language accepted by the grammar contains λ, then rule $S \rightarrow \lambda$ is permitted, where S is the start variable.

Any CFG can be transformed to a CNF with a finite sequence of steps. In this exercise, you'll use JFLAP to transform the following CFG to a CNF.

$S \rightarrow ASA|aB$
$A \rightarrow B|S$
$B \rightarrow b|\lambda$
$X \rightarrow A$

Input the above grammar into JFLAP, or load the file CFG1.jff. See Figure 1 below. Then select Convert:Transfrom Grammar.



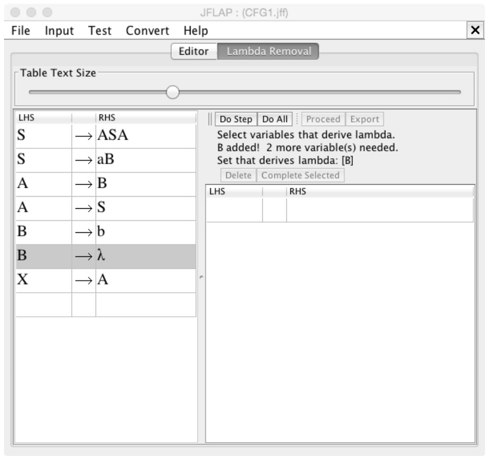
LHS		RHS
S	$\rightarrow$	ASA
S	$\rightarrow$	aB
A	$\rightarrow$	B
A	$\rightarrow$	S
B	$\rightarrow$	b
B	$\rightarrow$	λ
X	$\rightarrow$	A

Figure 1: An Example CFG

2 Transformation to Chomsky Normal Form

The transformation of a CFG to CNF is a series of steps in order. Each step involves removal of certain productions or addition of new ones so the the new set of productions generates the same language as the old set of productions. The steps are described below.

2.1 Removal of λ-Productions



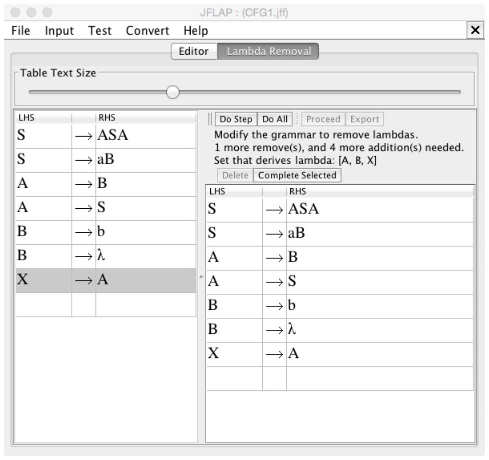
Select variables that derive lambda.
B added! 2 more variable(s) needed.
Set that derives lambda: [B]

LHS		RHS
S	→	ASA
S	→	aB
A	→	B
A	→	S
B	→	b
B	→	λ
X	→	A

LHS		RHS

Figure 2: Finding λ-Productions

Here we look for variables that derive λ. In the CFG above, the production $B \to \lambda$ is a λ-production. There may be other variables that derive λ. For example, the sequence of productions $A \to B, B \to \lambda$ shows that A derives λ. In Figure 2, we have selected the highlighted production $B \to \lambda$ since B derives λ. In the window on the right JFLAP informs you how many more variables that derive λ need to be selected. You can do this one step at a time or "Do All". See Figure 3.

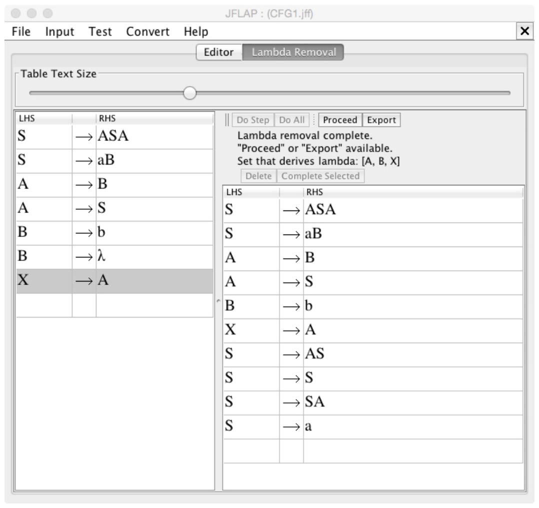


Modify the grammar to remove lambdas.
1 more remove(s), and 4 more addition(s) needed.
Set that derives lambda: [A, B, X]

LHS		RHS
S	→	ASA
S	→	aB
A	→	B
A	→	S
B	→	b
B	→	λ
X	→	A

LHS		RHS
S	→	ASA
S	→	aB
A	→	B
A	→	S
B	→	b
B	→	λ
X	→	A

Figure 3: Identifying λ-Productions



LHS		RHS
S	→	ASA
S	→	aB
A	→	B
A	→	S
B	→	b
B	→	λ
X	→	A

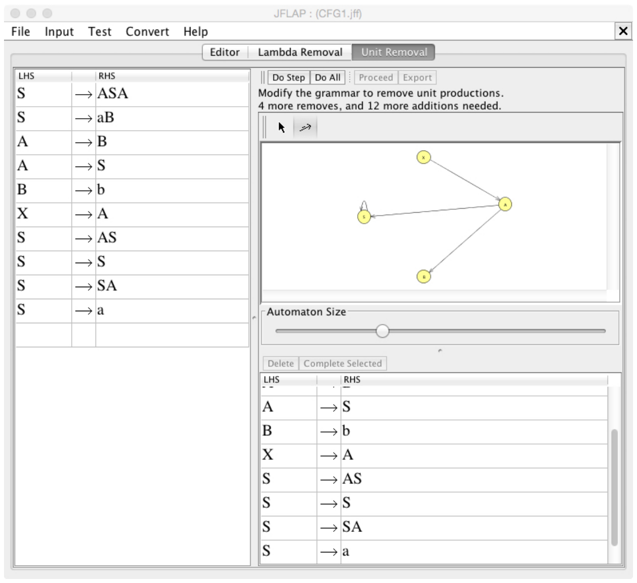
Lambda removal complete.
"Proceed" or "Export" available.
Set that derives lambda: [A, B, X]

LHS		RHS
S	→	ASA
S	→	aB
A	→	B
A	→	S
B	→	b
X	→	A
S	→	AS
S	→	S
S	→	SA
S	→	a

Figure 4: Result of λ-removal

Once all the λ-productions are identified, we remove such productions and modify some others so as to generate the same language. Here, we need to remove the production $B \to \lambda$. However, since the sequence of productions $S \to aB, B \to \lambda$ gives $S \to a$, we need to add this production. We similarly do this for all other variables that derive λ. This result is shown in Figure 4.

2.2 Removing Unit Productions



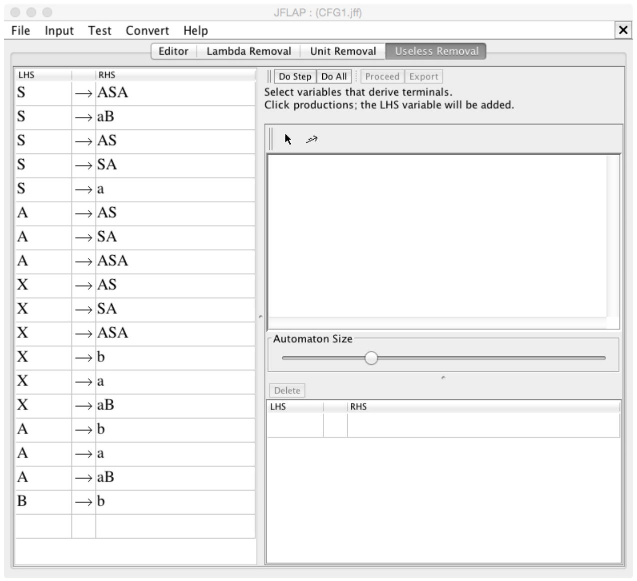
LHS		RHS
S	→	ASA
S	→	aB
A	→	B
A	→	S
B	→	b
X	→	A
S	→	AS
S	→	S
S	→	SA
S	→	a

Modify the grammar to remove unit productions.
4 more removes, and 12 more additions needed.

LHS		RHS
A	→	S
B	→	b
X	→	A
S	→	AS
S	→	S
S	→	SA
S	→	a

Figure 5: Unit removal

A *unit production* is a production of the form $C \to D$. Since a CNF does not permit such productions, they must be removed and some other appropriate productions added so that the language of the grammar stays the same. In our example, the unit productions are $A \to B, A \to S, X \to A$, and $S \to S$. See Figure 5. JFLAP works with a dependency graph as shown in Figure 5. The four unit productions are shown as arcs. For each pair of variables C and D in the dependency graph such that there is a directed path from C to D, and for each production $D \to w$, where w is string that is not a single variable, we add a production $C \to w$ to the grammar. Then we remove all the unit productions. See Figure 6.



LHS		RHS
S	$\longrightarrow$	ASA
S	$\longrightarrow$	aB
S	$\longrightarrow$	AS
S	$\longrightarrow$	SA
S	$\longrightarrow$	a
A	$\longrightarrow$	AS
A	$\longrightarrow$	SA
A	$\longrightarrow$	ASA
X	$\longrightarrow$	AS
X	$\longrightarrow$	SA
X	$\longrightarrow$	ASA
X	$\longrightarrow$	b
X	$\longrightarrow$	a
X	$\longrightarrow$	aB
A	$\longrightarrow$	b
A	$\longrightarrow$	a
A	$\longrightarrow$	aB
B	$\longrightarrow$	b

Select variables that derive terminals.
Click productions; the LHS variable will be added.

LHS		RHS

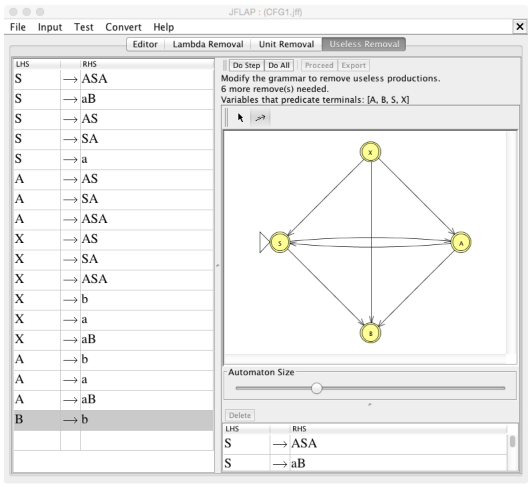
Figure 6: Unit removal completed

2.3 Removing Useless Productions

A production is *useless* if it cannot be used in a derivation. What productions in the grammar in Figure 6 are useless? JFLAP uses a dependency graph to determine useless productions. See Figure 7. The nodes of this dependency graph are the variables that derive terminals. An arc from C to D indicates that there is a C production that has D on the right side.

Now for every node C in the dependency graph that is not reachable from S by a directed path, all the C productions are useless and are removed. In our example, the X productions are useless and are removed. The result is shown in Figure 8.

We are now ready for the last step to complete our transformation to a CNF. In Figure 8, we observe that four productions are highlighted. These are precisely the productions that are not permitted in a CNF. The other transformations are all in CNF.



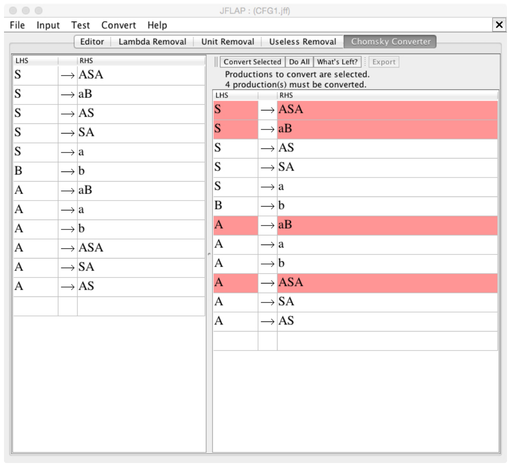
LHS		RHS
S	→	ASA
S	→	aB
S	→	AS
S	→	SA
S	→	a
A	→	AS
A	→	SA
A	→	ASA
X	→	AS
X	→	SA
X	→	ASA
X	→	b
X	→	a
X	→	aB
A	→	b
A	→	a
A	→	aB
B	→	b

Modify the grammar to remove useless productions.
6 more remove(s) needed.
Variables that predicate terminals: [A, B, S, X]

LHS		RHS
S	→	ASA
S	→	aB

Figure 7: Identifying useless productions

2.4 Converting to CNF



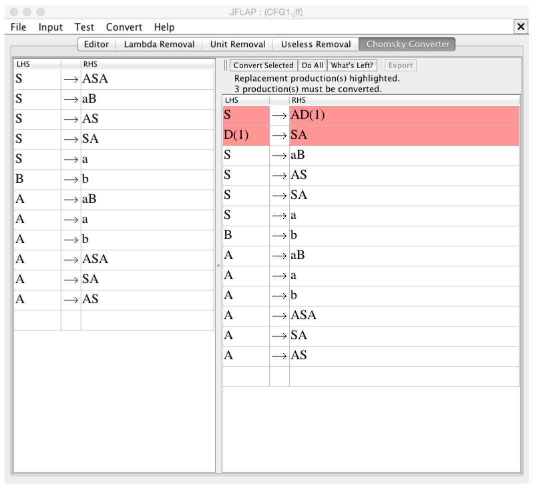
LHS		RHS
S	→	ASA
S	→	aB
S	→	AS
S	→	SA
S	→	a
B	→	b
A	→	aB
A	→	a
A	→	b
A	→	ASA
A	→	SA
A	→	AS

Productions to convert are selected.
4 production(s) must be converted.

LHS		RHS
S	→	ASA
S	→	aB
S	→	AS
S	→	SA
S	→	a
B	→	b
A	→	aB
A	→	a
A	→	b
A	→	ASA
A	→	SA
A	→	AS

Figure 8: Useless productions removed

We must now remove each of the four highlighted productions in Figure 8 and add appropriate new productions. For example, the production $S \to ASA$ is replaced by $S \to AD(1)$ and $D(1) \to SA$, where $D(1)$ is a new variable. See Figure 9. When this process is repeated for the remining three productions, the CNF is achieved as shown in Figure 10.

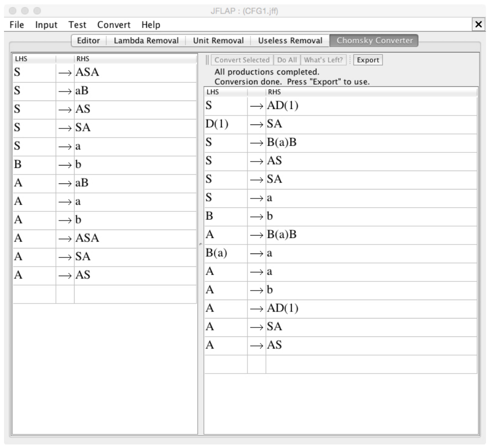

JFLAP : (CFG1.jff)
File Input Test Convert Help
Editor Lambda Removal Unit Removal Useless Removal Chomsky Converter
Convert Selected Do All What's Left? Export

LHS		RHS
S	→	ASA
S	→	aB
S	→	AS
S	→	SA
S	→	a
B	→	b
A	→	aB
A	→	a
A	→	b
A	→	ASA
A	→	SA
A	→	AS

Replacement production(s) highlighted.
3 production(s) must be converted.

LHS		RHS
S	→	AD(1)
D(1)	→	SA
S	→	aB
S	→	AS
S	→	SA
S	→	a
B	→	b
A	→	aB
A	→	a
A	→	b
A	→	ASA
A	→	SA
A	→	AS

Figure 9: Modifying productions for CNF


JFLAP : (CFG1.jff)
File Input Test Convert Help
Editor Lambda Removal Unit Removal Useless Removal Chomsky Converter
Convert Selected Do All What's Left? Export

LHS		RHS
S	→	ASA
S	→	aB
S	→	AS
S	→	SA
S	→	a
B	→	b
A	→	aB
A	→	a
A	→	b
A	→	ASA
A	→	SA
A	→	AS

All productions completed.
Conversion done. Press "Export" to use.

LHS		RHS
S	→	AD(1)
D(1)	→	SA
S	→	B(a)B
S	→	AS
S	→	SA
S	→	a
B	→	b
A	→	B(a)B
B(a)	→	a
A	→	a
A	→	b
A	→	AD(1)
A	→	SA
A	→	AS

Figure 10: CNF transformation completed

3 Additional Exercise

Repeat the above process to transform the following grammar to CNF.
$S \rightarrow BSB|B|\lambda$
$B \rightarrow bb|\lambda$

4 References

1. Introduction to the Theory of Computation (Third Edition), Michael Sipser. Cengage Learning. 2013.

2. JFLAP - An Interactive Formal Languages and Automata Package, Susan H. Rodger and Thomas W Finley. Jones and Bartlett Publishers. 2006