

Converting a CFG to Chomsky Normal Form_{JP}

- Prerequisite knowledge:
Context-Free Languages
Context-Free Grammars
JFLAP Tutorial

In this unit, we will look at the process of converting a Context Free Grammar into an equivalent grammar in Chomsky Normal Form.

With no restrictions on the form of the right-hand sides of CFG grammar rules, some grammars can be hard to use; that is, some tasks and algorithms can be difficult to carry out or inefficient. For example, if the grammar has ϵ -productions, brute-force parsing can be slow. Since there exist multiple equivalent CFG expressions of a given language, we consider the potential to rewrite the grammar into a form that has more desirable properties. A grammar is said to be in a *normal form* if it cannot be rewritten any further under the specified rules of transformation.

Given a CFG grammar G , we are looking for an alternative CFG F that is a transformed version of G (produces the same language) but for which some tasks of interest are easier to perform. One example of a useful CFG normal form is Greibach Normal Form (GNF) established by Sheila Greibach. A CFG is in GNF if the right-hand sides of all production rules start with a terminal symbol, optionally followed by some variables, and the only permissible ϵ -production is from the start symbol. That is, all rules have the form $\mathbf{A} \rightarrow \mathbf{a}\mathbf{B}$ or $\mathbf{S} \rightarrow \epsilon$ where $\mathbf{A} \square$ Variables (nonterminals), $\mathbf{a} \square$ Terminals, $\mathbf{B} \square$ Variables*, and \mathbf{S} is the start symbol. In every derivation produced by a GNF grammar exactly one terminal is generated for each rule application. A grammar in GNF is easily converted to a PDA with no ϵ -transitions, which are guaranteed to halt.

We will now consider another CFG normal form, namely, Chomsky Normal Form.

Description of Chomsky Normal Form (CNF)

A CFG is said to be in Chomsky Normal Form (established by Noam Chomsky) if all production rules have the form $\mathbf{A} \rightarrow \mathbf{BC}$, $\mathbf{A} \rightarrow \mathbf{a}$, or $\mathbf{S} \rightarrow \epsilon$ where $\mathbf{A}, \mathbf{B}, \mathbf{C} \square$ Variables (nonterminals), $\mathbf{a} \square$ Terminals, and \mathbf{S} is the start symbol.

Consider the following context-free grammar. (See: CFG2CNF.jff)

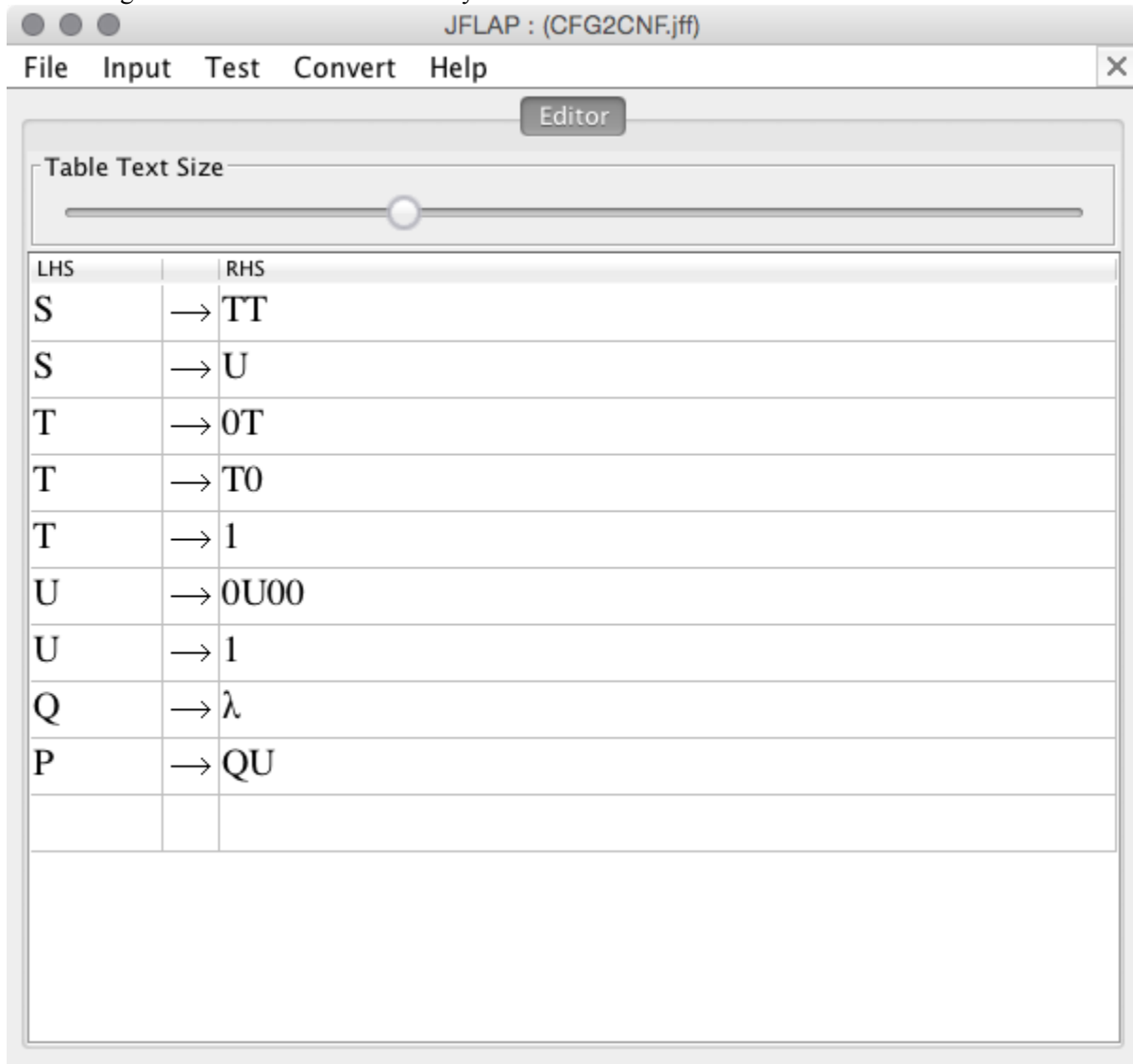
Variables = { P, Q, S, T, U }
Terminals = { 0, 1 }
Start Symbol = S

$S \rightarrow TT$
 $S \rightarrow U$
 $T \rightarrow 0T$
 $T \rightarrow T0$
 $T \rightarrow 1$
 $U \rightarrow 0U00$
 $U \rightarrow 1$
 $Q \rightarrow \epsilon$
 $P \rightarrow QU$

This CFG is not in CNF because it violates several of the conditions. For example, $Q \rightarrow \epsilon$ is invalid because Q is not the start symbol. $S \rightarrow U$ is invalid because there is only one variable on the right-hand side.

Identify all of the violations of CNF in this grammar.

Enter this grammar into JFLAP and verify that it is context free.



JFLAP : (CFG2CNF.jff)

File Input Test Convert Help

Editor

Table Text Size

LHS		RHS
S	→	TT
S	→	U
T	→	0T
T	→	T0
T	→	1
U	→	0U00
U	→	1
Q	→	λ
P	→	QU

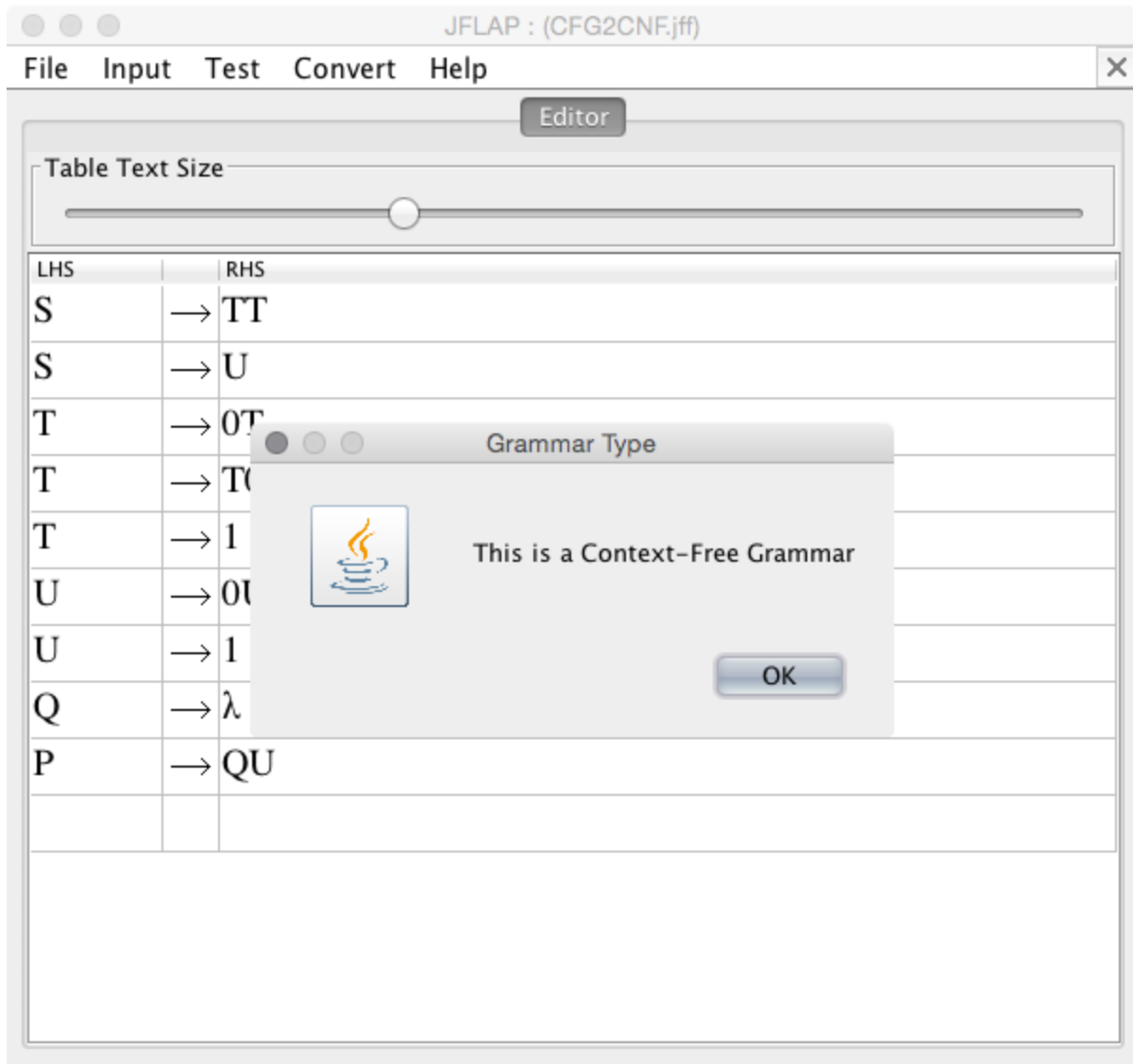
JFLAP : (CFG2CNF.jff)

File Input **Test** Convert Help

Test for Grammar Type **tor**

Table Text Size

LHS		RHS
S	→	TT
S	→	U
T	→	0T
T	→	T0
T	→	1
U	→	0U00
U	→	1
Q	→	λ
P	→	QU

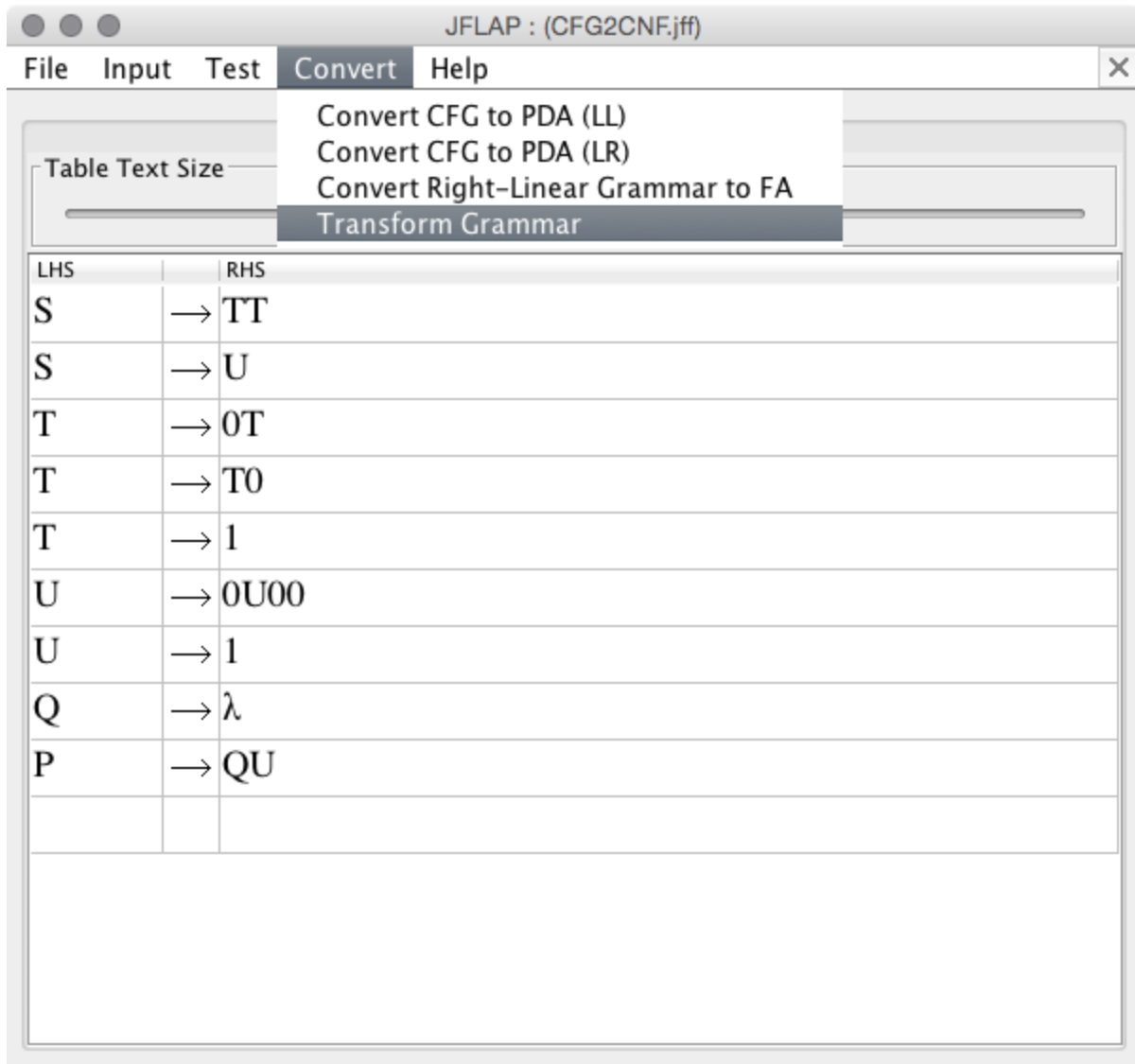


Example of the CFG → CNF Conversion Process

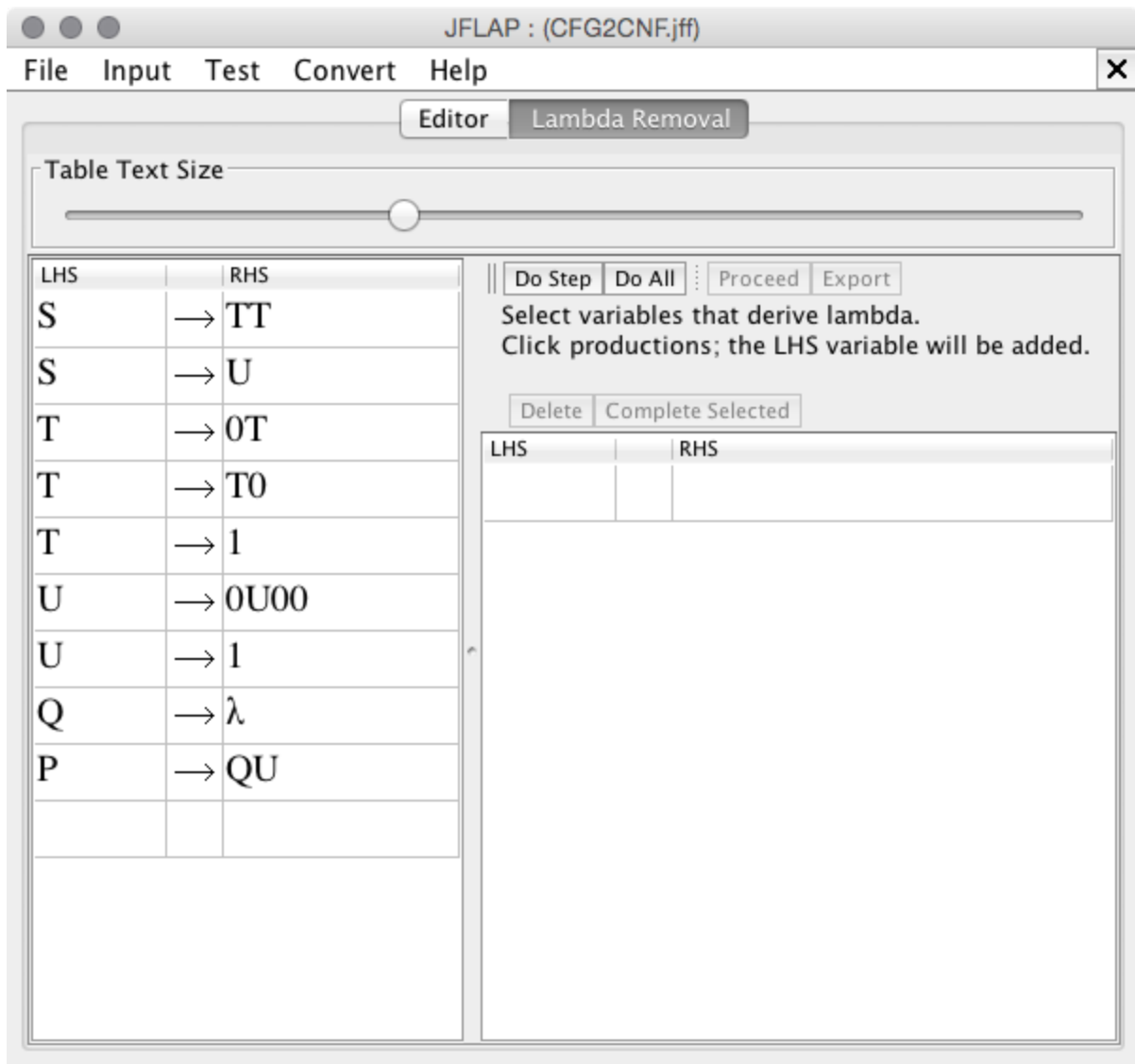
One approach to converting a CFG into an equivalent grammar in CNF is to successively replace objects in the CFG to get closer to the requirements for CNF while maintaining the integrity of the language recognized.

The following sequence follows a path through the conversion process as provided by JFLAP.

Begin by selecting *Convert > Transform Grammar*.



The first step in this transformation involves the removal of ϵ -productions (Lambda removal).



The conversion process begins by choosing *Do Step*.

Table Text Size

LHS	RHS
S	→ TT
S	→ U
T	→ 0T
T	→ T0
T	→ 1
U	→ 0U00
U	→ 1
Q	→ λ
P	→ QU

|| Do Step Do All ... Proceed Export

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

Delete Complete Selected

LHS	RHS
S	→ TT
S	→ U
T	→ 0T
T	→ T0
T	→ 1
U	→ 0U00
U	→ 1
Q	→ λ
P	→ QU

Select the rule that produces lambda and select *Delete*.

JFLAP : (CFG2CNF.jff)

File Input Test Convert Help

Editor Lambda Removal

Table Text Size

LHS	RHS
S	→ TT
S	→ U
T	→ 0T
T	→ T0
T	→ 1
U	→ 0U00
U	→ 1
Q	→ λ
P	→ QU

|| Do Step Do All ... Proceed Export

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

Delete Complete Selected

LHS	RHS
S	→ TT
S	→ U
T	→ 0T
T	→ T0
T	→ 1
U	→ 0U00
U	→ 1
Q	→ λ
P	→ QU

JFLAP : (CFG2CNF.jff)

File Input Test Convert Help

Editor Lambda Removal

Table Text Size

LHS	RHS
S	→ TT
S	→ U
T	→ OT
T	→ T0
T	→ 1
U	→ OU00
U	→ 1
Q	→ λ
P	→ QU

|| Do Step Do All ... Proceed Export

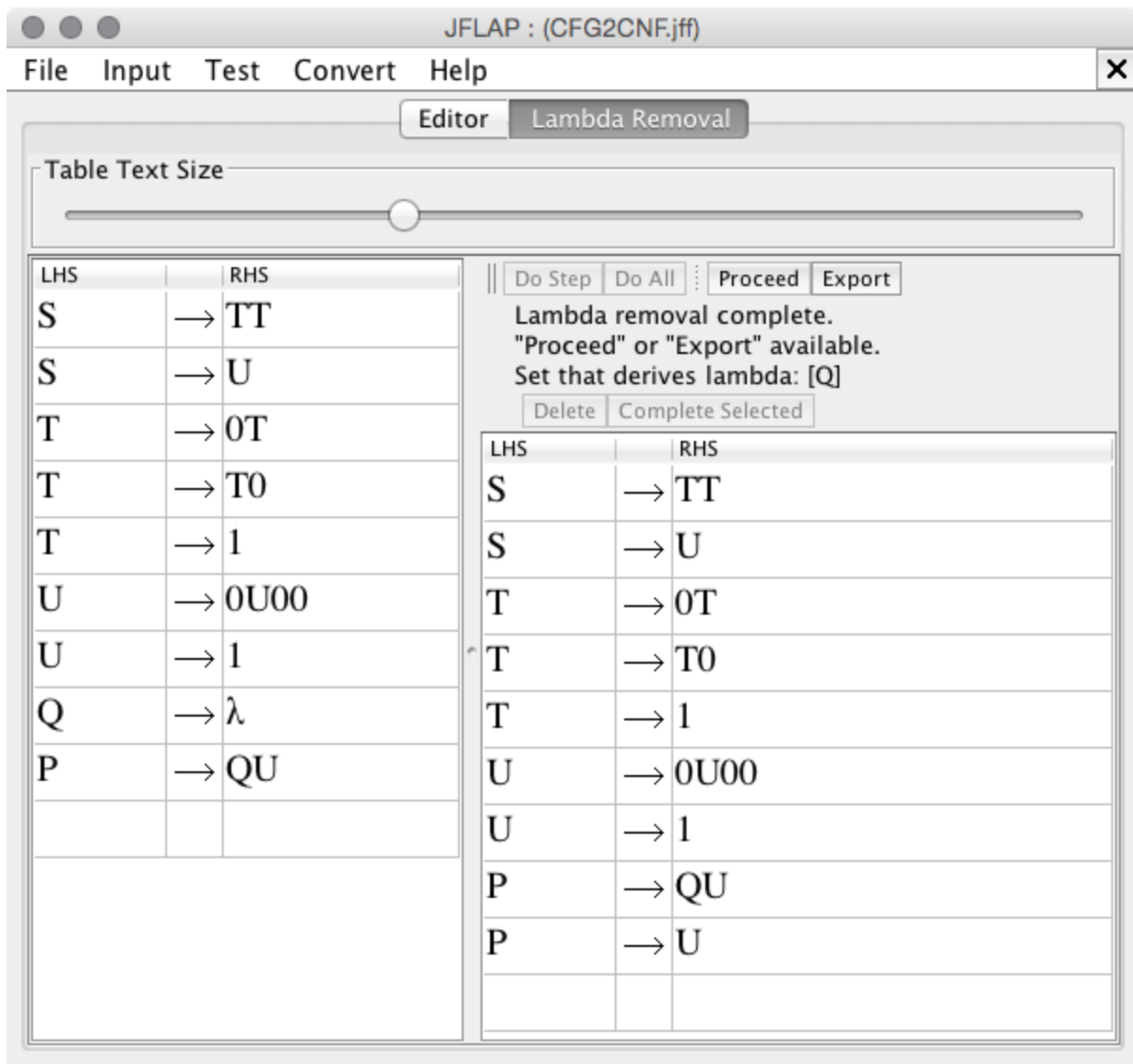
Modify the grammar to remove lambdas.
 0 more remove(s), and 1 more addition(s) needed.
 Set that derives lambda: [Q]

Delete Complete Selected

LHS	RHS
S	→ TT
S	→ U
T	→ OT
T	→ T0
T	→ 1
U	→ OU00
U	→ 1
P	→ QU

An additional rule is necessary to account for the lambda removal.

Select *Do Step*.



No additional Lambda removal is necessary.

Select *Proceed*. JFLAP will move into "Unit Removal" mode and provide a state-transition visualization.

JFLAP : (CFG2CNF.jff)

File Input Test Convert Help

Editor Lambda Removal Unit Removal

LHS	RHS
S	→ TT
S	→ U
T	→ 0T
T	→ T0
T	→ 1
U	→ 0U00
U	→ 1
P	→ QU
P	→ U

|| Do Step Do All ... Proceed Export

Complete unit production visualization.
2 more transition(s) needed.

Automaton Size

Delete Complete Selected

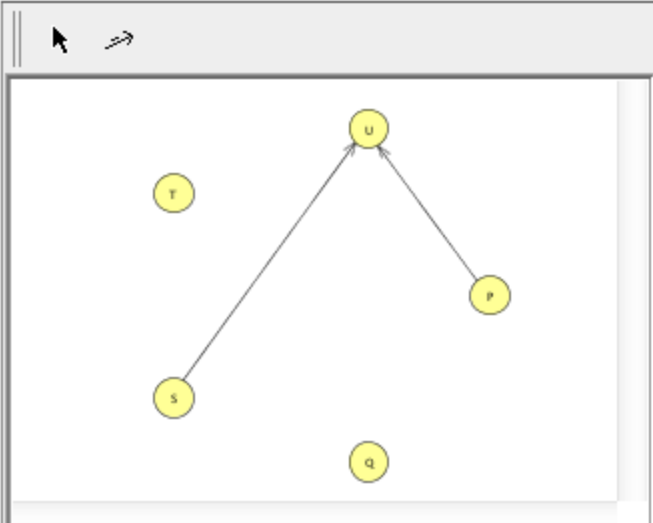
LHS	RHS

Select *Do Step* twice to follow along with the unit-removal transformation.

LHS	RHS
S	→ TT
S	→ U
T	→ 0T
T	→ T0
T	→ 1
U	→ 0U00
U	→ 1
P	→ QU
P	→ U

|| Do Step Do All ... Proceed Export

Modify the grammar to remove unit productions. 2 more removes, and 4 more additions needed.



Automaton Size

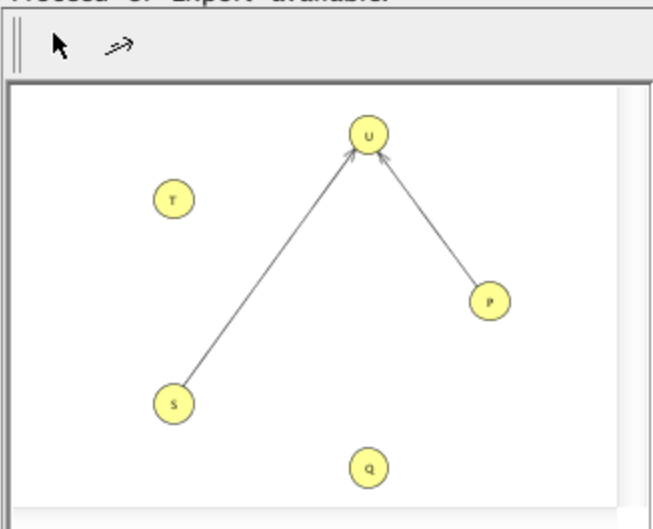
Delete Complete Selected

LHS	RHS
S	→ TT
S	→ U
T	→ 0T
T	→ T0
T	→ 1
U	→ 0U00
U	→ 1
P	→ QU
P	→ U

LHS	RHS
S	→ TT
S	→ U
T	→ 0T
T	→ T0
T	→ 1
U	→ 0U00
U	→ 1
P	→ QU
P	→ U

|| Do Step Do All Proceed Export

Unit removal complete.
"Proceed" or "Export" available.



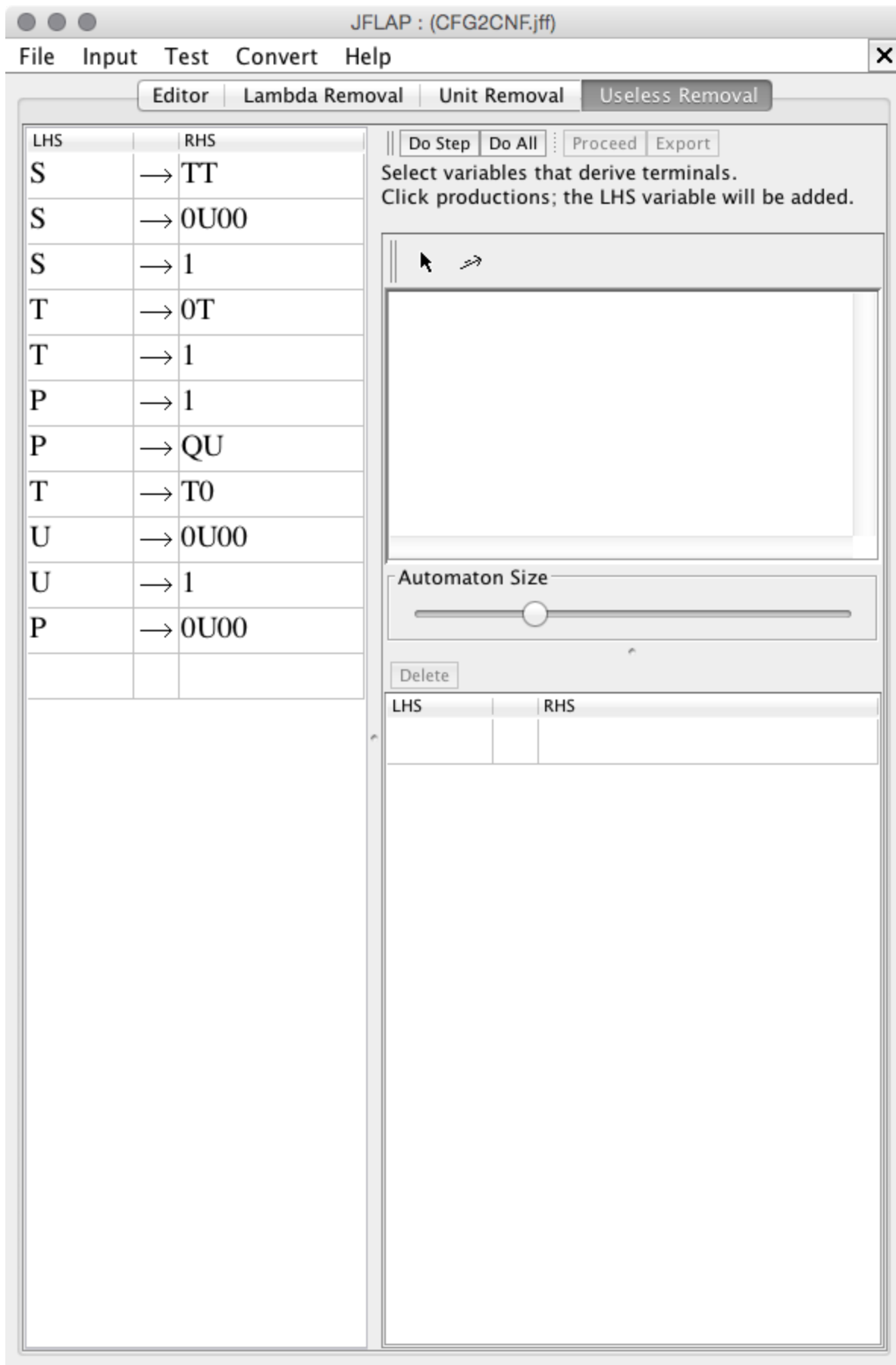
Automaton Size

—————○—————

Delete Complete Selected

LHS	RHS
S	→ TT
T	→ 0T
T	→ T0
T	→ 1
U	→ 0U00
U	→ 1
P	→ QU
P	→ 1
S	→ 0U00
P	→ 0U00
S	→ 1

Unit removal is complete. Select *Proceed* to begin removing useless productions.



Select *Do Step* three times to follow along with the process of identifying and eliminating useless variables and productions.

JFLAP : (CFG2CNF.jff)

File Input Test Convert Help

Editor Lambda Removal Unit Removal Useless Removal

LHS	RHS
S	→ TT
S	→ 0U00
S	→ 1
T	→ 0T
T	→ 1
P	→ 1
P	→ QU
T	→ T0
U	→ 0U00
U	→ 1
P	→ 0U00

Do Step Do All Proceed Export

Complete dependency graph.
3 more transition(s) needed.
Variables that predicate terminals: [P, S, T, U]

Automaton Size

Delete

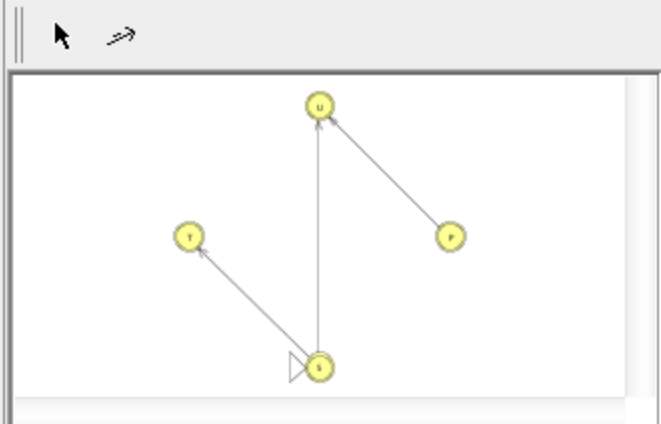
LHS	RHS
S	→ TT
S	→ 0U00
S	→ 1
T	→ 0T
T	→ 1
P	→ 1
T	→ T0
U	→ 0U00
U	→ 1
P	→ 0U00

LHS	RHS
S	→ TT
S	→ 0U00
S	→ 1
T	→ 0T
T	→ 1
P	→ 1
P	→ QU
T	→ T0
U	→ 0U00
U	→ 1
P	→ 0U00

|| Do Step Do All ... Proceed Export

Modify the grammar to remove useless productions.
2 more remove(s) needed.

Variables that predicate terminals: [P, S, T, U]



Automaton Size

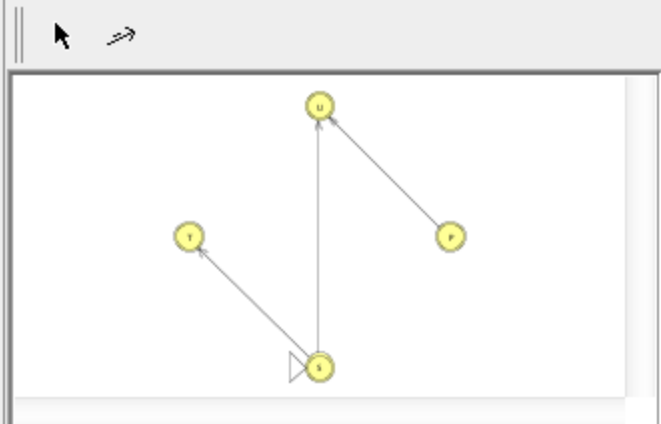
Delete

LHS	RHS
S	→ TT
S	→ 0U00
S	→ 1
T	→ 0T
T	→ 1
P	→ 1
T	→ T0
U	→ 0U00
U	→ 1
P	→ 0U00

LHS	RHS
S	→ TT
S	→ 0U00
S	→ 1
T	→ 0T
T	→ 1
P	→ 1
P	→ QU
T	→ T0
U	→ 0U00
U	→ 1
P	→ 0U00

|| Do Step Do All ... Proceed Export

Useless removal complete.
 "Proceed" or "Export" available.
 Variables that predicate terminals: [P, S, T, U]



Automaton Size

Delete

LHS	RHS
S	→ TT
S	→ 0U00
S	→ 1
T	→ 0T
T	→ 1
T	→ T0
U	→ 0U00
U	→ 1

All useless variables and products have been removed. Select *Proceed* to enter the final phase, Chomsky Converter.

JFLAP : (CFG2CNF.jff)

File Input Test Convert Help

Editor Lambda Removal Unit Removal Useless Removal Chomsky Converter

LHS		RHS
S	→	TT
S	→	OU00
S	→	1
T	→	0T
T	→	1
T	→	T0
U	→	OU00
U	→	1

Convert Selected Do All What's Left? Export

Welcome to the Chomsky converter.
4 production(s) must be converted.

LHS		RHS
S	→	TT
S	→	OU00
S	→	1
T	→	0T
T	→	1
T	→	T0
U	→	OU00
U	→	1

Note that 4 of the productions must be converted.

Which productions require conversion?

Select one of those production rules and choose *Convert Selected*.

JFLAP : (CFG2CNF,jff)

File Input Test Convert Help

Editor Lambda Removal Unit Removal Useless Removal Chomsky Converter

LHS		RHS
S	→	TT
S	→	OU00
S	→	1
T	→	0T
T	→	1
T	→	T0
U	→	OU00
U	→	1

Convert Selected Do All What's Left? Export

Welcome to the Chomsky converter.
4 production(s) must be converted.

LHS		RHS
S	→	TT
S	→	OU00
S	→	1
T	→	0T
T	→	1
T	→	T0
U	→	OU00
U	→	1

LHS	RHS
S	→ TT
S	→ 0U00
S	→ 1
T	→ 0T
T	→ 1
T	→ T0
U	→ 0U00
U	→ 1

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

LHS	RHS
S	→ TT
S	→ B(0)UB(0)B(0)
B(0)	→ 0
S	→ 1
T	→ 0T
T	→ 1
T	→ T0
U	→ 0U00
U	→ 1

Note that a single step in the conversion process does not necessarily reduce the number of productions that need to be converted. Select *What's Left?* to highlight rules that need to be converted.

JFLAP : (CFG2CNF.jff)

File Input Test Convert Help

Editor Lambda Removal Unit Removal Useless Removal Chomsky Converter

LHS	RHS
S	→ TT
S	→ 0U00
S	→ 1
T	→ 0T
T	→ 1
T	→ T0
U	→ 0U00
U	→ 1

Convert Selected Do All What's Left? Export

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

LHS	RHS
S	→ TT
S	→ B(0)UB(0)B(0)
B(0)	→ 0
S	→ 1
T	→ 0T
T	→ 1
T	→ T0
U	→ 0U00
U	→ 1

Continue selecting and converting productions until all appropriate productions have been converted.

JFLAP : (CFG2CNF.jff)

File Input Test Convert Help

Editor Lambda Removal Unit Removal Useless Removal Chomsky Converter

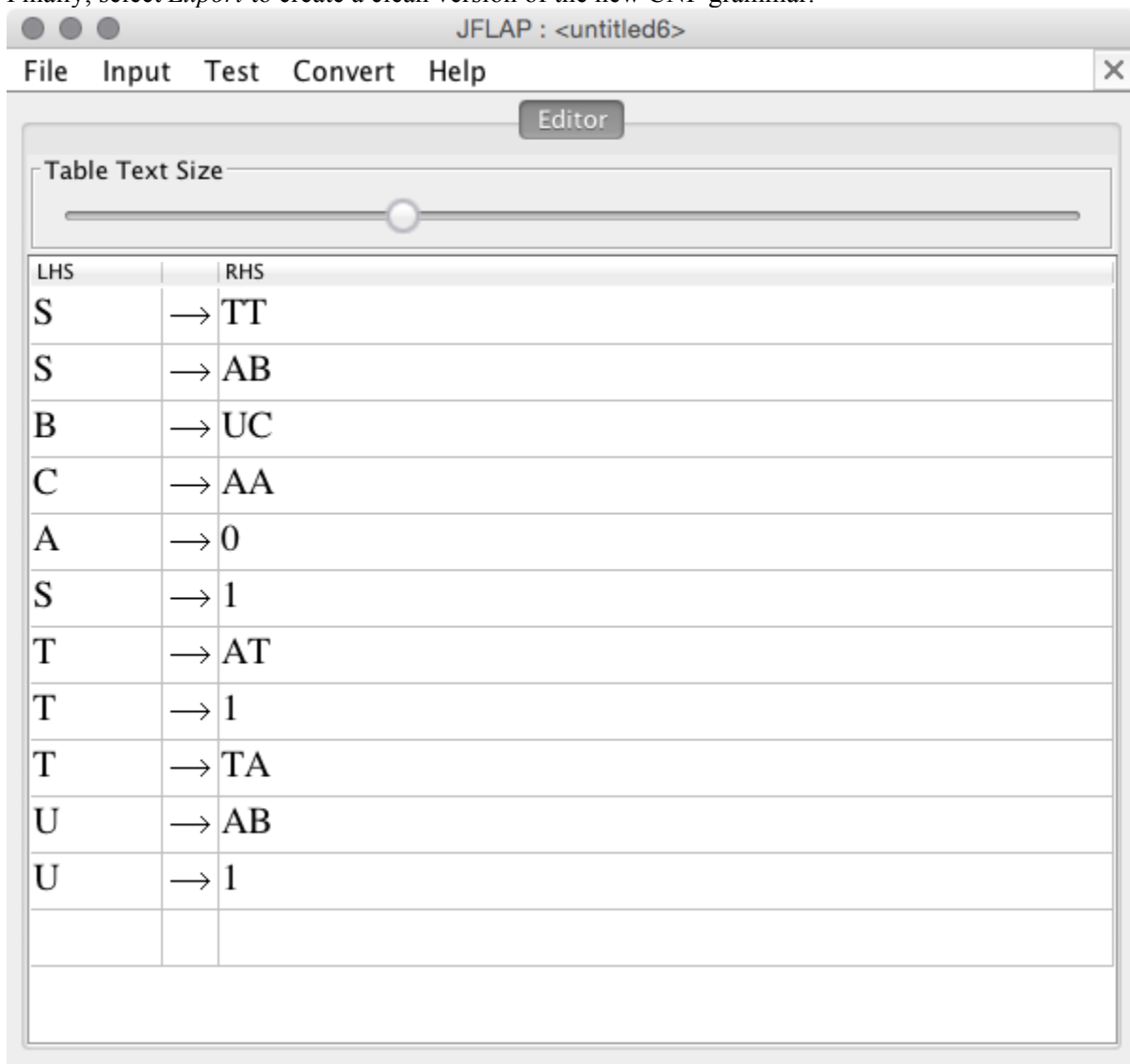
LHS	RHS
S	→ TT
S	→ 0U00
S	→ 1
T	→ 0T
T	→ 1
T	→ T0
U	→ 0U00
U	→ 1

Convert Selected Do All What's Left? Export

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

LHS	RHS
S	→ TT
S	→ B(0)D(1)
D(1)	→ UD(2)
D(2)	→ B(0)B(0)
B(0)	→ 0
S	→ 1
T	→ B(0)T
T	→ 1
T	→ TB(0)
U	→ B(0)D(1)
U	→ 1

Finally, select *Export* to create a clean version of the new CNF grammar.




The screenshot shows the JFLAP Editor window with the following table of productions:

LHS		RHS
S	→	TT
S	→	AB
B	→	UC
C	→	AA
A	→	0
S	→	1
T	→	AT
T	→	1
T	→	TA
U	→	AB
U	→	1

Verify that every production is in the appropriate form for a CNF grammar.

Use *Test > Test for Grammar Type* as additional verification.

Test for Grammar Type **tor**

Table Text Size 

LHS		RHS
S	→	TT
S	→	AB
B	→	UC
C	→	AA
A	→	0
S	→	1
T	→	AT
T	→	1
T	→	TA
U	→	AB
U	→	1

JFLAP : <untitled6>


File Input Test Convert Help

Editor

Table Text Size

LHS		RHS
S	→	TT
S	→	AB
B	→	UC
C		
A		
S		
T		
T		
T	→	TA
U	→	AB
U	→	1

Grammar Type

 This is a CNF Grammar (also Context-Free Grammar)

OK

Questions to Think About

1. How can you convince yourself that this CNF grammar is equivalent to the original CFG?
Answer: The steps of conversion constitute a proof.
2. Convert the CNF grammar to a PDA. Run test inputs that demonstrate the same behavior of the PDA with the original grammar.