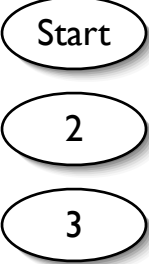
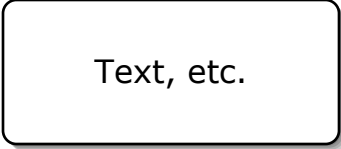
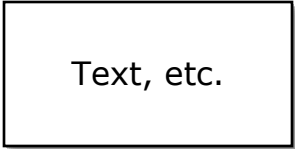
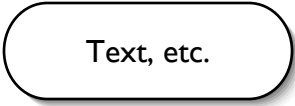
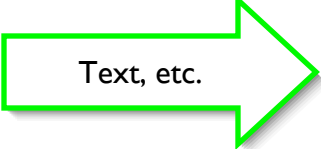
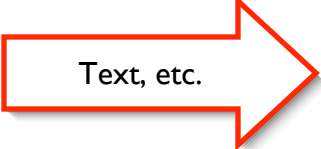
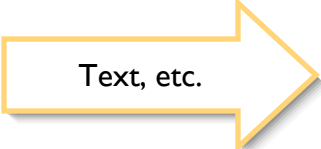




Legend:

---

State markers:	
Main user visible content and interface:	
Tree construction/ argumentation area:	
Control:	
Correct move:	
Incorrect move:	
Incomplete move:	
Navigational move:	
Feedback content:	

---

Start

Try to construct the parse tree for the following expression in order to determine whether or not it is a formula.

Start by selecting the main operator of an expression and creating the appropriate number of branches. Then fill in the subexpressions at the ends of those branches.

Once you reach a node containing an expression that cannot be further decomposed by any syntactic rules, classify that expression as either an atomic formula, by pressing the "Atomic" button when that node is selected, or as not well-formed, by pressing the "Not Well-Formed" button when the node is selected. Once all the terminal nodes have been classified correctly, you'll have completed the exercise.

$( (\forall x)( R(e) \ \& \ \neg G(f,g) ) \vee (\exists x)( R(e) \ \& \ \neg F(g,f) ) )$

Create binary branch

Create unary branch

Hint

Atomic

Not Well-Formed

Create binary branch

Create unary branch

Atomic/ Not Well-Formed

Hint



2

3

5

Hints

2

Main operator selected, and  
is a binary connective

[Create branches.]

Binary connective selected,  
but is not main operator

The [name of connective] selected is not  
the main operator of that expression.

Unary operator selected  
and is main operator

The selected [negation/quantifier] is the  
main operator of that expression, but it is a  
unary operator, not a binary one.

Unary operator selected,  
but is not main operator

The selected [negation/quantifier] is neither  
the main operator of that expression, nor is  
it a binary operator.

Atomic formula selected

You have selected an atomic formula.

Nothing selected

[Prompt user to make a selection.]

3

Main operator selected, and  
is negation or a quantifier

[Create branch.]

Unary operator selected,  
but is not main operator

The selected [negation/quantifier] is not the  
main operator of that expression.

Binary connective selected,  
and is main operator

The [name of connective] selected is the  
main operator of that expression, but  
[name of connective] is a binary connective,  
not a unary one.

Binary connective selected,  
but is not main operator

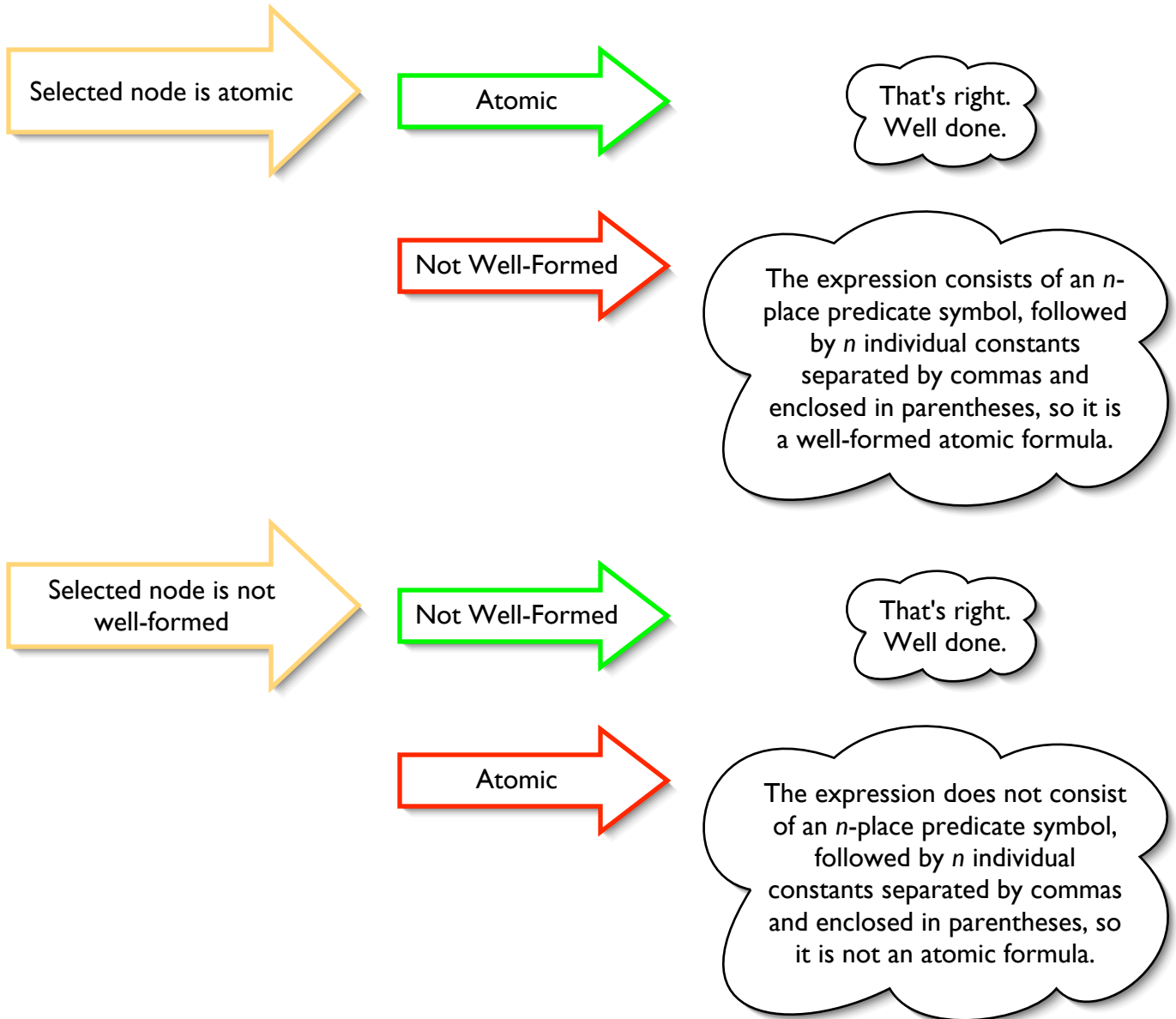
The [name of connective] selected is  
neither the main operator of that  
expression, nor a unary operator.

Atomic formula selected

You have selected an atomic formula.

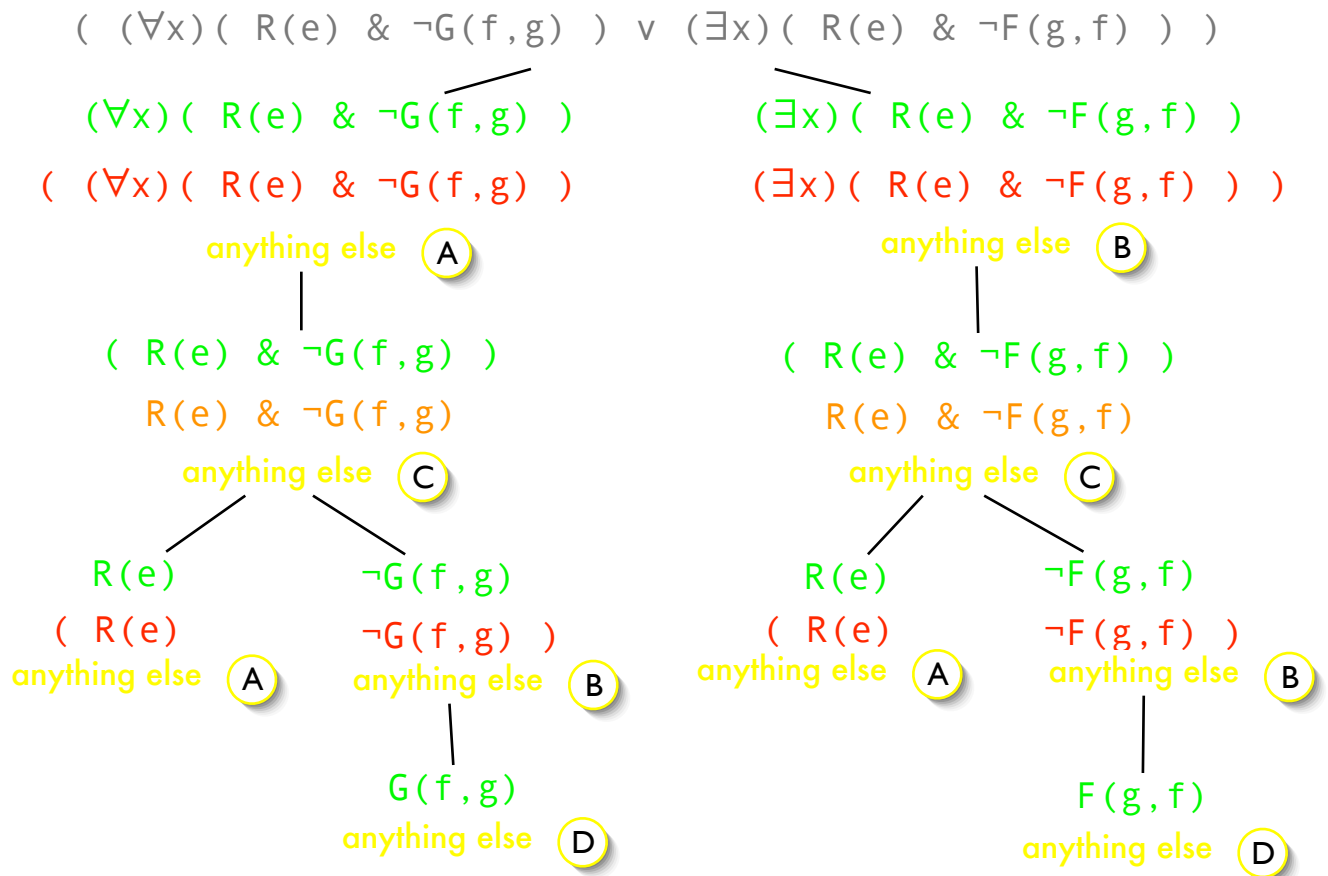
Nothing selected

[Prompt user to make a selection.]



Feedback for entering expressions at nodes:

Key:



Feedback:

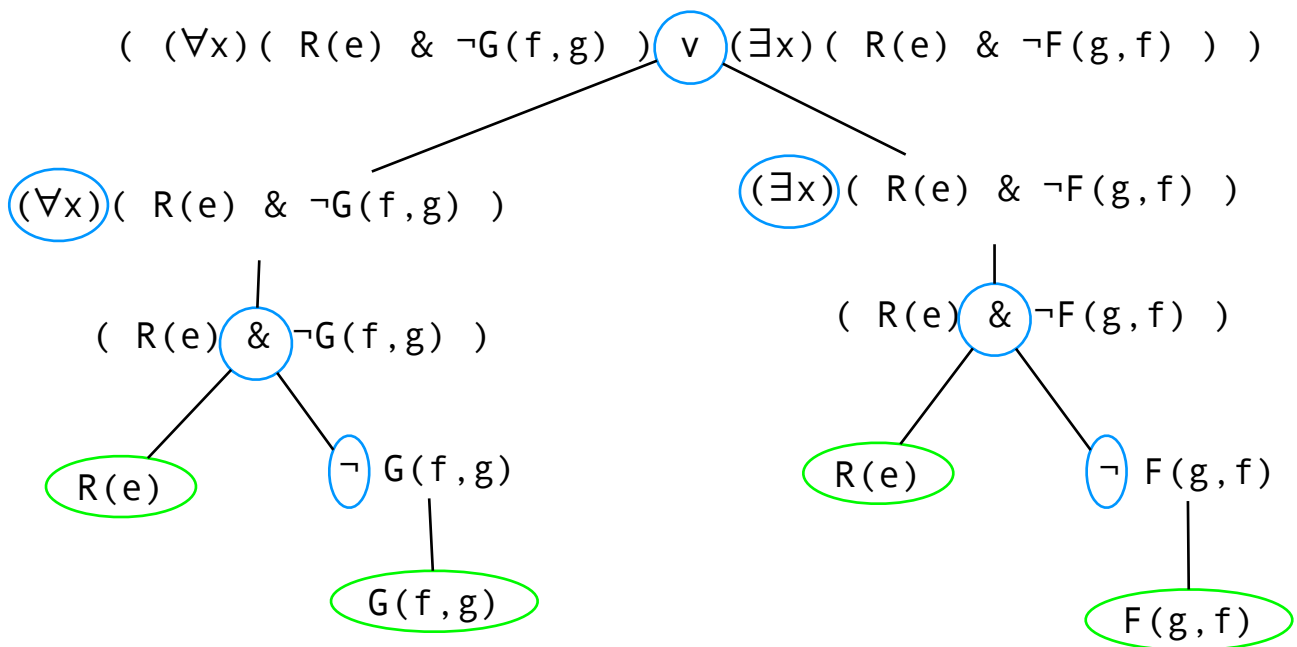
That's right!

Don't forget that the outermost parentheses are added by the application of a syntactic rule, so the outermost parentheses of an expression higher in the tree will not appear again in the branches below.

We never omit outermost parentheses in parse trees, but other than that you have the right formula.

- (A) For a binary branch, the left-hand subexpression will always consist of that portion of the original expression between the leftmost outer parenthesis and the connective that was added by the application of the syntactic rule.
- (B) For a binary branch, the right-hand subexpression will always consist of that portion of the original expression between the rightmost outer parenthesis and the connective that was added by the application of the syntactic rule.
- (C) For a unary branch corresponding to an application of one of the quantifier rules, the subexpression will always consist of the original expression minus the outermost quantifier that was added by the application of the syntactic rule.
- (D) For a unary branch corresponding to an application of the rule for negation, the subexpression will always consist of the original expression minus the leftmost negation that was added by the application of the syntactic rule.

Solution:



Atomic terminal nodes are circled in green, and non-well-formed nodes in red (in this case, all the terminal nodes are atomic). Additionally, the main connective of each expression is circled in blue.

Recall that **all** terminal nodes must be classified correctly by the user before the activity is complete.

For reference:

Operator Name	Symbol	Type
Conjunction	&	Binary
Disjunction	∨	Binary
Conditional	→	Binary
Negation	¬	Unary
Universal Quantifier	(∀x)	Unary
Existential Quantifier	(∃x)	Unary

## Hints

Click [here](#) to get help on how to construct the tree.

Click [here](#) to view the syntactic rules and parse tree rules.

Links should be to the following files, respectively:

parsetreeconstruction4help.gif  
parsetreeconstruction4hint.gif

The latter is already done, but I'll wait on the help images until after the interface has been finalized.



v (top node)

Start by selecting the main operator of the formula.

If the leftmost symbol of the expression is a negation, then that is the main connective, and if the expression has a quantifier at its far left, then that is the main operator. If not, look for a binary connective with a parenthesis to either side (a right parenthesis on its left, and a left parenthesis on its right).

In this expression, the disjunction is the main connective.

Click [here](#) to highlight the connective.



$(\forall x)( R(e) \ \& \ \neg G(f,g) )$   
 $(\exists x)( R(e) \ \& \ \neg F(g,f) )$

textbox

The expression that should go at the end of a binary branch is just that portion of the parent expression that comes between the outermost parenthesis on the same side as the branch and the connective itself.

For this branch, the parent expression is  $( (\forall x)( R(e) \ \& \ \neg G(f,g) ) \vee (\exists x)( R(e) \ \& \ \neg F(g,f) ) )$ , so the expression that should go at the end of this branch is the portion of that between the [left/right] outer parenthesis and the disjunction.

The expression you should enter here is

$(\forall x)( R(e) \ \& \ \neg G(f,g) )$   
 $(\exists x)( R(e) \ \& \ \neg F(g,f) )$ .

operator

If an expression has a quantifier at its far left, with no parenthesis to the left of the quantifier's left parenthesis, then the quantifier is the main operator for the expression..

In this expression, the quantifier is the main connective.

Click [here](#) to highlight the connective

$( R(e) \ \& \ \neg G(f,g) )$   
 $( R(e) \ \& \ \neg F(g,f) )$

textbox

The expression that should go at the end of a unary branch corresponding to the application of one of the syntactic rules for the quantifiers is just the portion of the original expression remaining after the quantifier is removed.

For this branch, the parent expression is  
 $(\forall x)( R(e) \ \& \ \neg G(f,g) )$   
 $(\exists x)( R(e) \ \& \ \neg F(g,f) )$ ,  
so the expression that should go at the end of this branch is the portion of the expression remaining after removing the quantifier.

The expression you should enter here is

$( R(e) \ \& \ \neg G(f,g) )$   
 $( R(e) \ \& \ \neg F(g,f) )$ .

&

If only a single binary connective occurs in an expression, then as long as the expression is enclosed in parentheses, that binary connective is the one to select.

This expression is enclosed in parentheses, and the only binary connective it contains is a single occurrence of conjunction.

In this expression, the conjunction is the main connective.

Click [here](#) to highlight the connective

$R(e)$

textbox

The expression that should go at the end of a binary branch is just that portion of the parent expression that comes between the outermost parenthesis on the same side as the branch and the connective itself.

For this branch, the parent expression is  
 $( R(e) \& \neg G(f,g) )$   
 $( R(e) \& \neg F(g,f) )$ ,  
so the expression that should go at the end of this branch is the portion of that between the left outer parenthesis and the conjunction.

The expression you should enter here is  
 $R(e)$ .

$R(e)$

If there are no occurrences of any connectives or quantifiers in an expression, then the expression cannot be further decomposed according to the syntactic rules.

There are no occurrences of connectives in  $R(e)$  so this expression cannot be further decomposed and should be classified as either atomic or not well-formed.

Since  $R(e)$  consists of a predicate letter followed by a single term enclosed in parentheses, it is a well-formed atomic formula.

$\neg G(f, g)$   
 $\neg F(g, f)$

textbox

The expression that should go at the end of a binary branch is just that portion of the parent expression that comes between the outermost parenthesis on the same side as the branch and the connective itself.

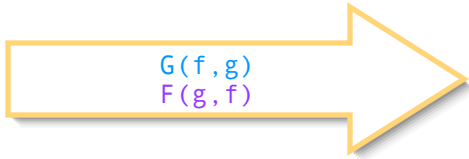
For this branch, the parent expression is  
 $( R(e) \& \neg G(f, g) )$   
 $( R(e) \& \neg F(g, f) )$ ,  
so the expression that should go at the end of this branch is the portion of that between the right outer parenthesis and the conjunction.

The expression you should enter here is  
 $\neg G(f, g)$   
 $\neg F(g, f)$ .

$\neg G(f, g)$   
 $\neg F(g, f)$

If the leftmost symbol in an expression is a negation symbol, then the expression could have been produced by an application of the syntactic rule for negation.

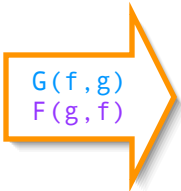
In this expression, the negation is the main connective.  
Click [here](#) to highlight the connective.



The expression that should go at the end of a unary branch corresponding to an application of the syntactic rule for negation is just that portion of the parent expression that remains after removing the leftmost negation symbol.

For this branch, the parent expression is  
 $\neg G(f, g)$   
 $\neg F(g, f)$ ,  
so the expression that should go at the end of this branch is the portion that remains after removing the leftmost negation symbol.

The expression you should enter here is  
 $G(f, g)$   
 $F(g, f)$ .



If there are no occurrences of any connectives or quantifiers in an expression, then the expression cannot be further decomposed according to the syntactic rules.

There are no occurrences of connectives in  $G(f, g)/F(g, f)$  so this expression cannot be further decomposed and should be classified as either atomic or not well-formed.

Since  $G(f, g)/F(g, f)$  consists of a predicate letter followed by a single term enclosed in parentheses, it is a well-formed atomic formula.