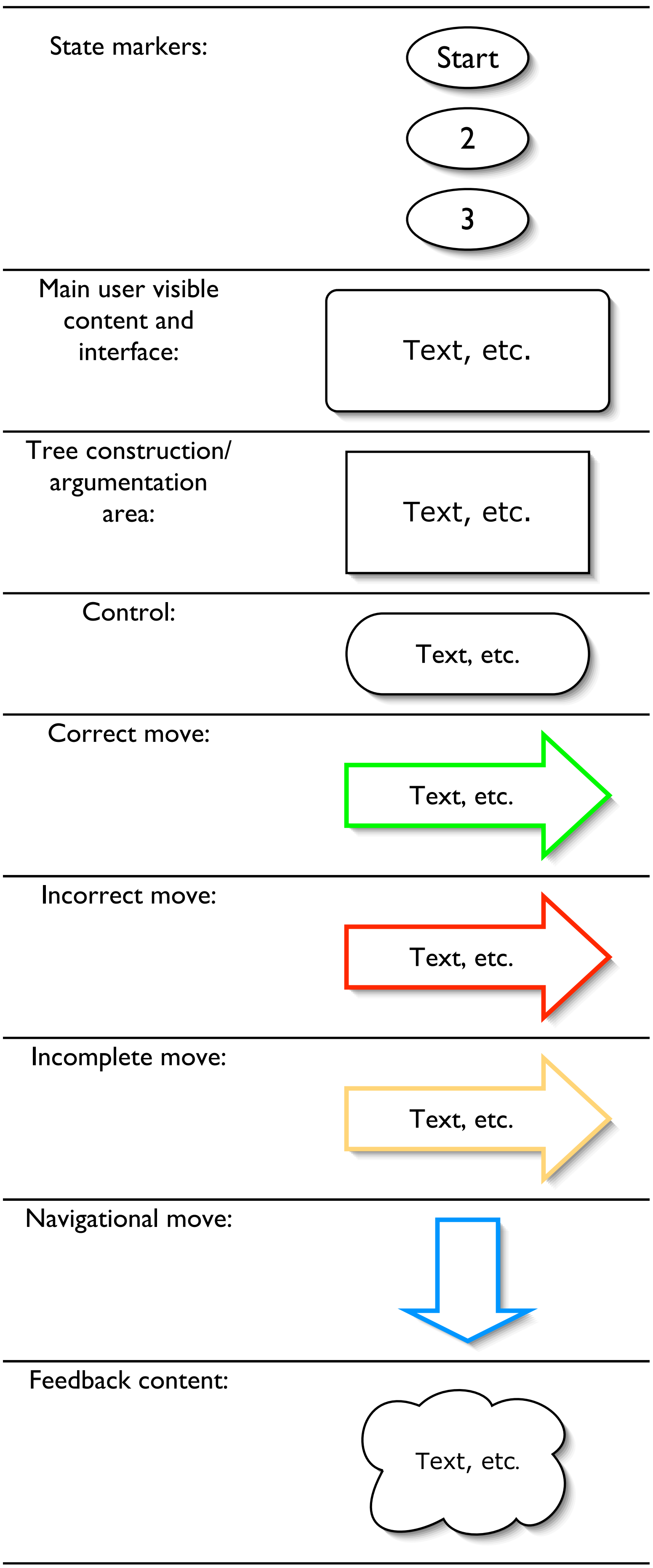
Legend:



State markers:

Start

2

3

Main user visible content and interface:

Text, etc.

Tree construction/ argumentation area:

Text, etc.

Control:

Text, etc.

Correct move:

Text, etc.

Incorrect move:

Text, etc.

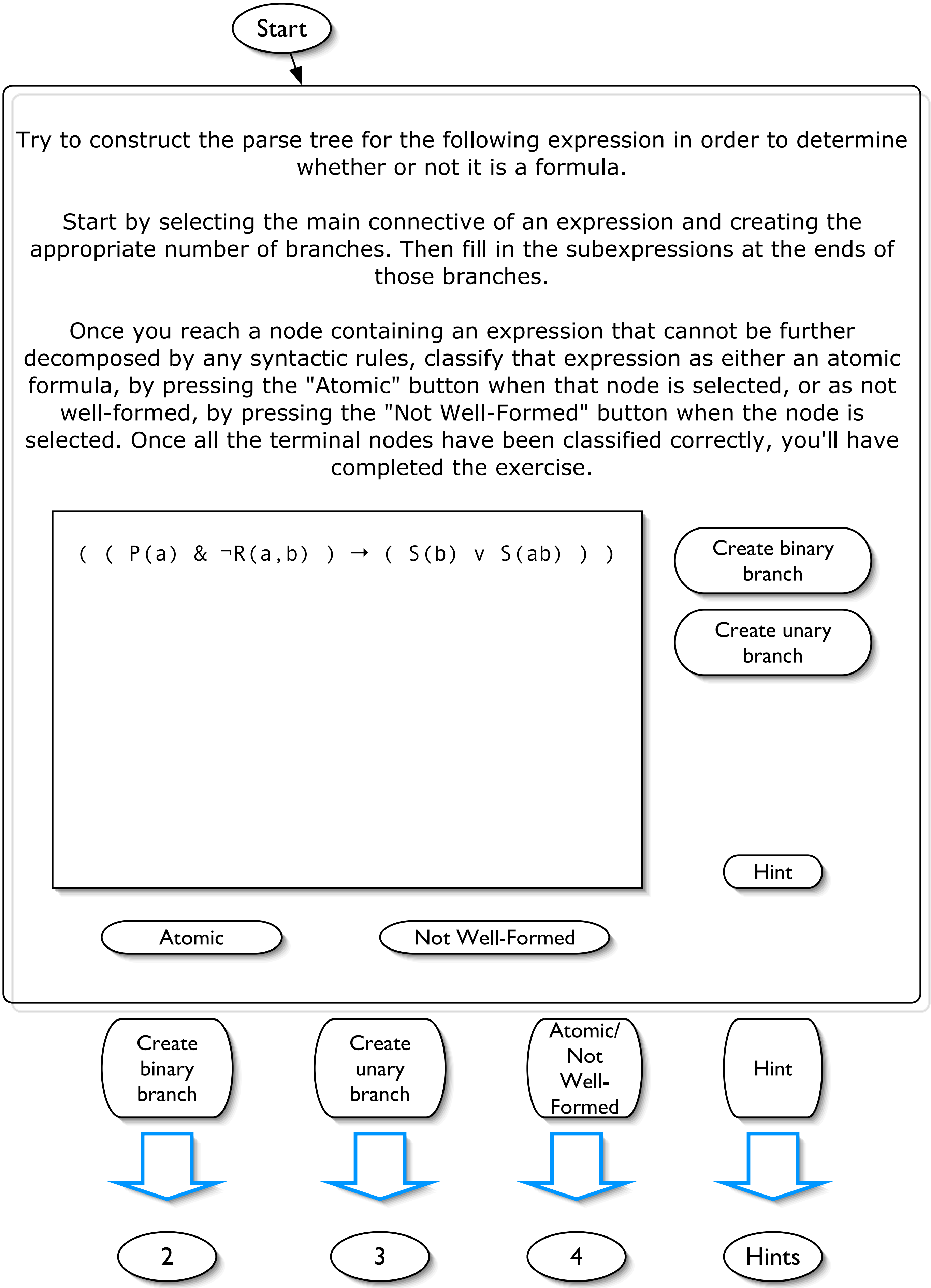
Incomplete move:

Text, etc.

Navigational move:

Feedback content:

Text, etc.



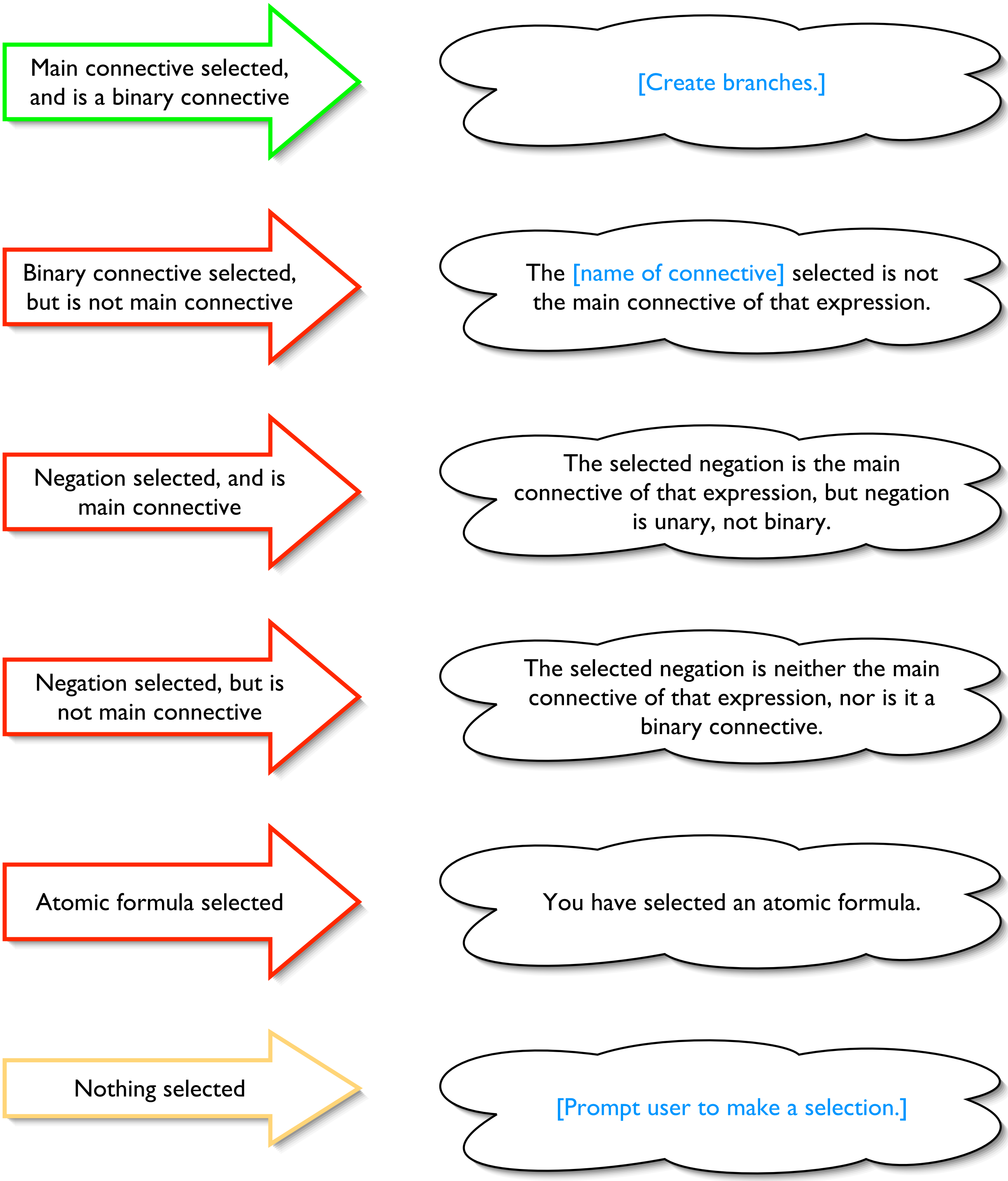
Try to construct the parse tree for the following expression in order to determine whether or not it is a formula.

Start by selecting the main connective 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.

((P(a) & ¬R(a,b)) → (S(b) v S(ab)))

2



Main connective selected, and is a binary connective

[Create branches.]

Binary connective selected, but is not main connective

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

Negation selected, and is main connective

The selected negation is the main connective of that expression, but negation is unary, not binary.

Negation selected, but is not main connective

The selected negation is neither the main connective of that expression, nor is it a binary connective.

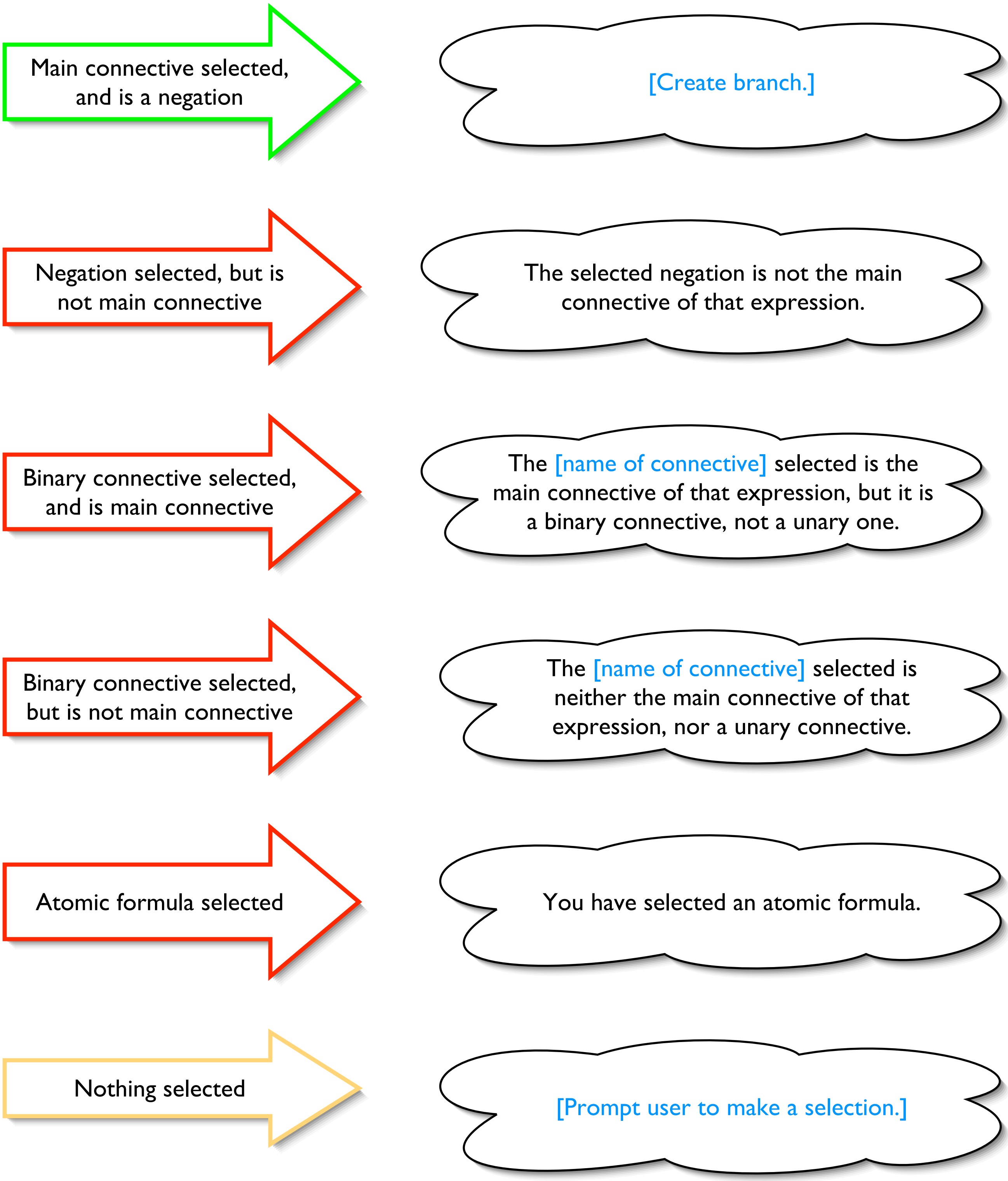
Atomic formula selected

You have selected an atomic formula.

Nothing selected

[Prompt user to make a selection.]

3



Main connective selected, and is a negation → [Create branch.]

Negation selected, but is not main connective → The selected negation is not the main connective of that expression.

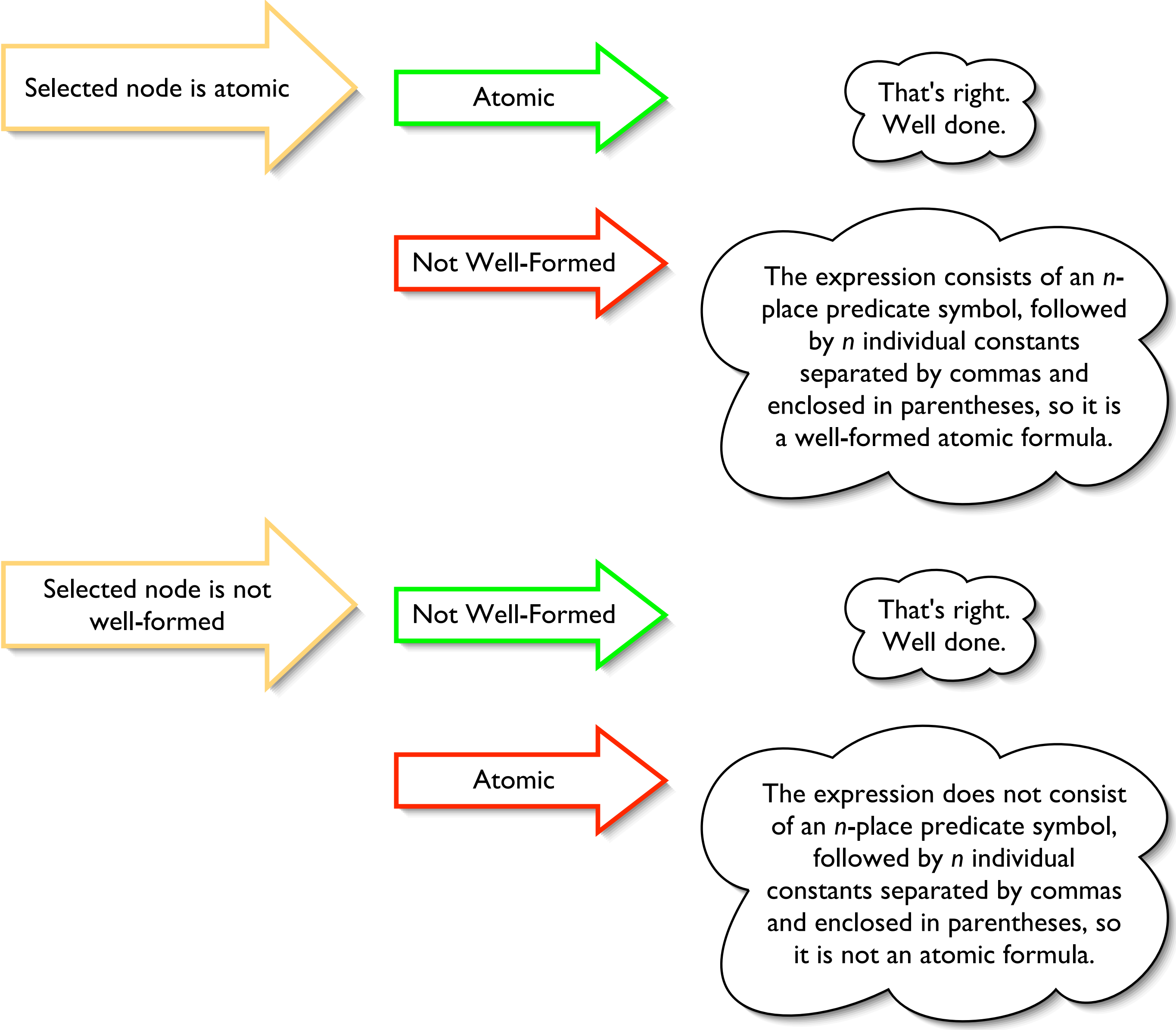
Binary connective selected, and is main connective → The [name of connective] selected is the main connective of that expression, but it is a binary connective, not a unary one.

Binary connective selected, but is not main connective → The [name of connective] selected is neither the main connective of that expression, nor a unary connective.

Atomic formula selected → You have selected an atomic formula.

Nothing selected → [Prompt user to make a selection.]

4



Selected node is atomic

Atomic

That's right. Well done.

Not Well-Formed

The expression consists of an *n*-place predicate symbol, followed by *n* individual constants separated by commas and enclosed in parentheses, so it is a well-formed atomic formula.

Selected node is not well-formed

Not Well-Formed

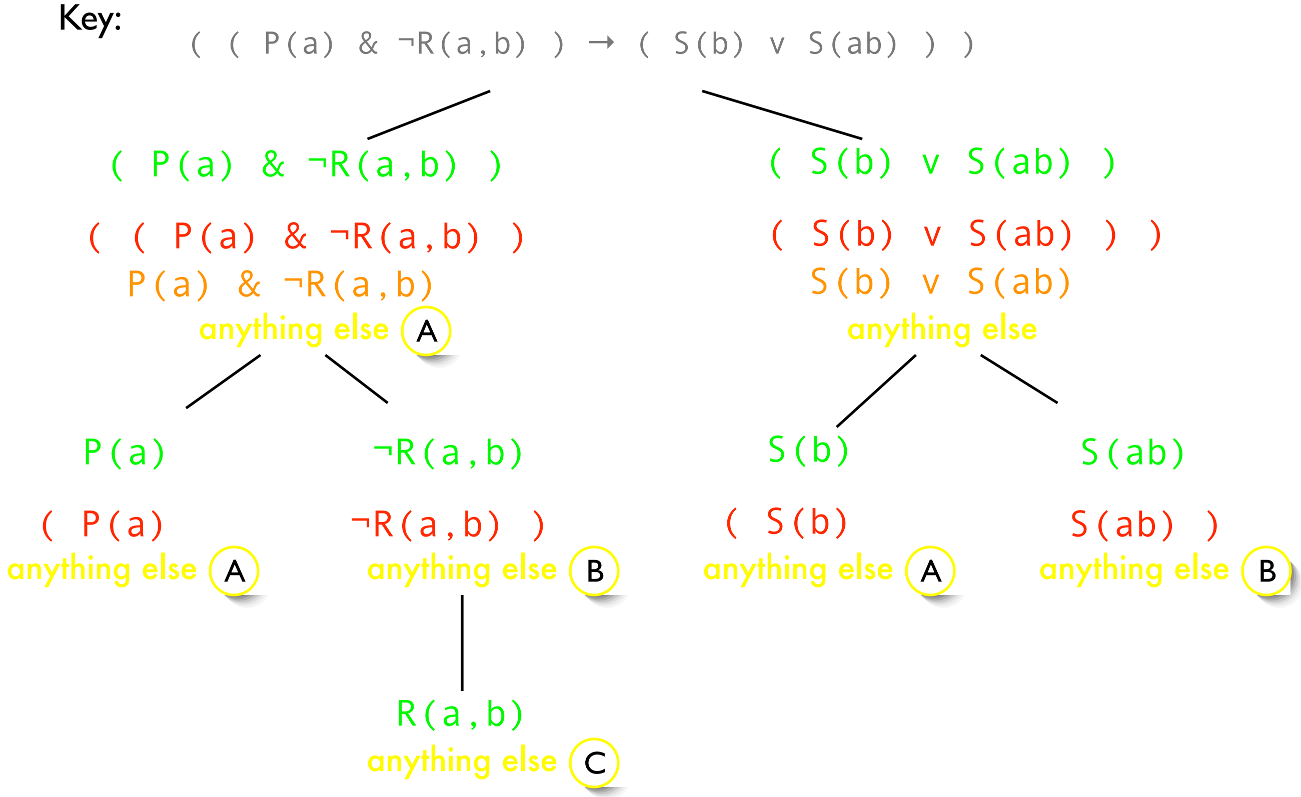
That's right. Well done.

Atomic

The expression does not consist of an *n*-place predicate symbol, followed by *n* individual constants separated by commas and enclosed in parentheses, so it is not an atomic formula.

Feedback for entering expressions at nodes:

Key:



((P(a) & ¬R(a,b)) → (S(b) v S(ab)))

(P(a) & ¬R(a,b))

((P(a) & ¬R(a,b))

P(a) & ¬R(a,b)

anything else A

(S(b) v S(ab))

(S(b) v S(ab)))

S(b) v S(ab)

anything else

P(a)

(P(a)

anything else A

¬R(a,b)

¬R(a,b))

anything else B

R(a,b)

anything else C

S(b)

(S(b)

anything else A

S(ab)

S(ab))

anything else B

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.

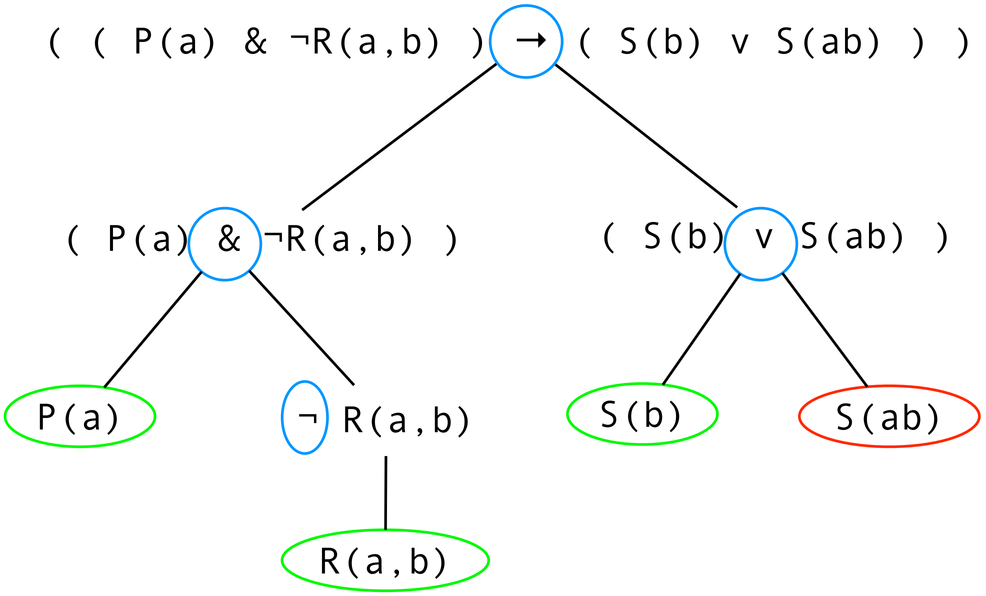
We never omit outermost parentheses in parse trees, but other than that you have the right formula.

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, 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. 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:

Connective Name	Symbol	Type
Conjunction	&	Binary
Disjunction	v	Binary
Conditional	→	Binary
Negation	¬	Unary

Hints

Click here to get help on how to construct the tree.

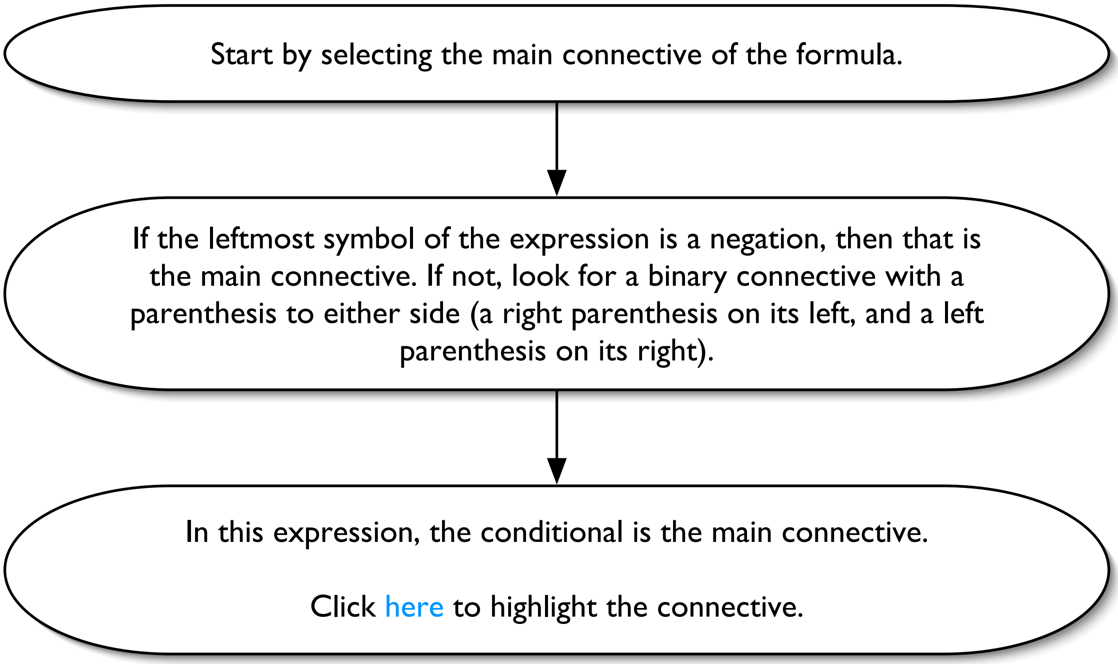
Click here to view the syntactic rules
and parse tree rules.

Links should be to the following files, respectively:

parsetreeconstruction3help.gif
parsetreeconstruction3hint.gif

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

→ (top node)

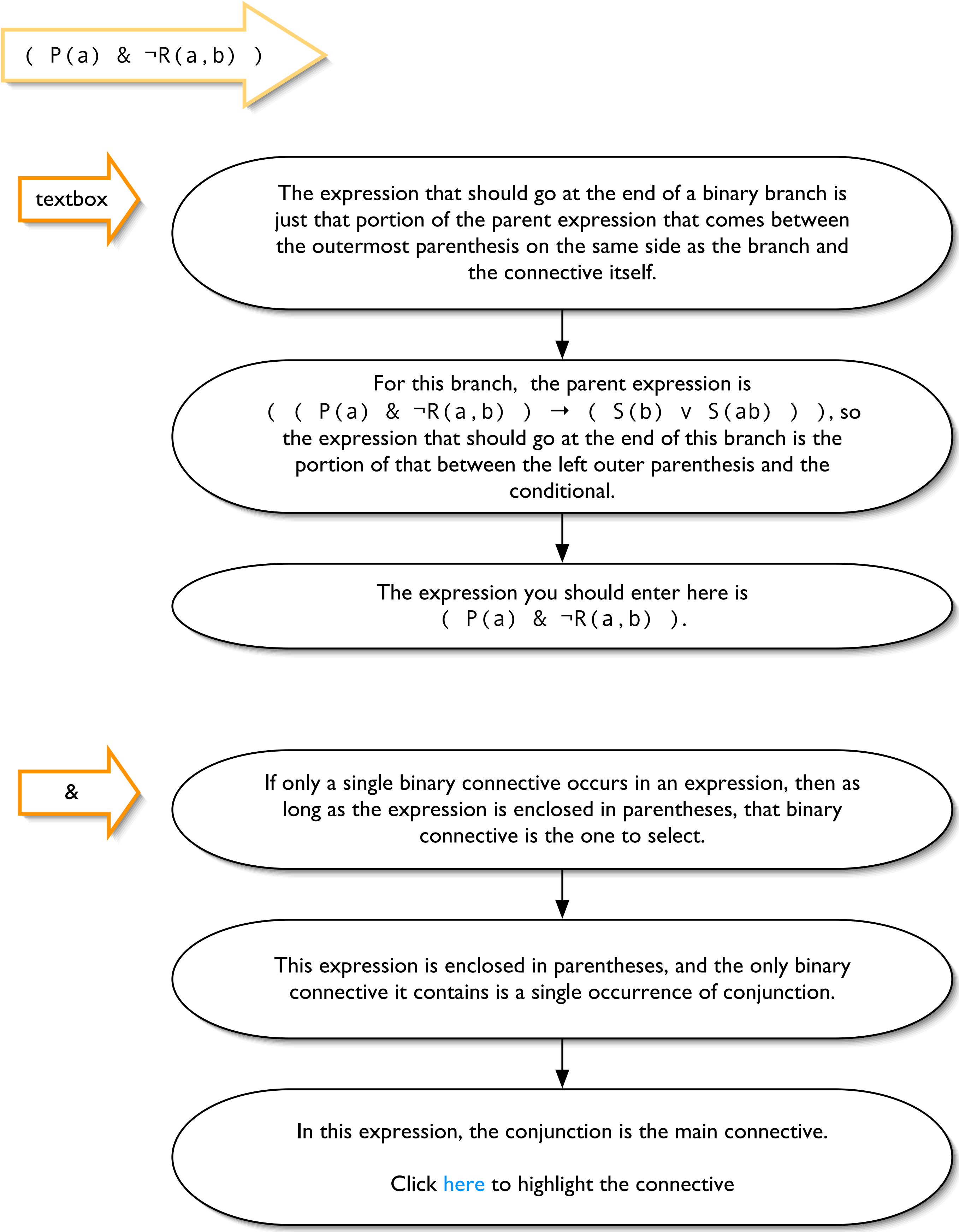


Start by selecting the main connective of the formula.

If the leftmost symbol of the expression is a negation, then that is the main connective. 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 conditional is the main connective.

Click here to highlight the connective.



(P(a) & ¬R(a,b))

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 ((P(a) & ¬R(a,b)) → (S(b) v S(ab))), so the expression that should go at the end of this branch is the portion of that between the left outer parenthesis and the conditional.

The expression you should enter here is (P(a) & ¬R(a,b)).

&

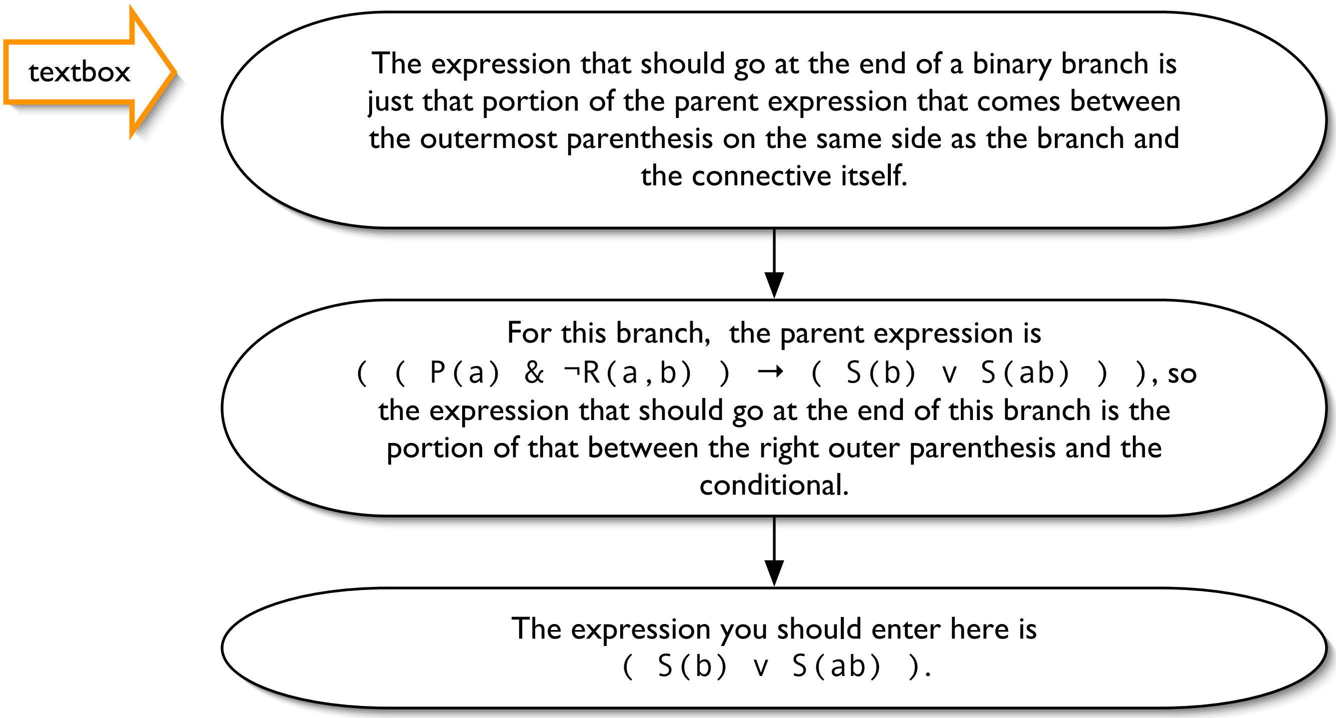
If only a single binary connective occurs in an expression, then as long as the expression is enclosed in parentheses, that binary connective is the one to select.

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

(S(b) v S(ab))

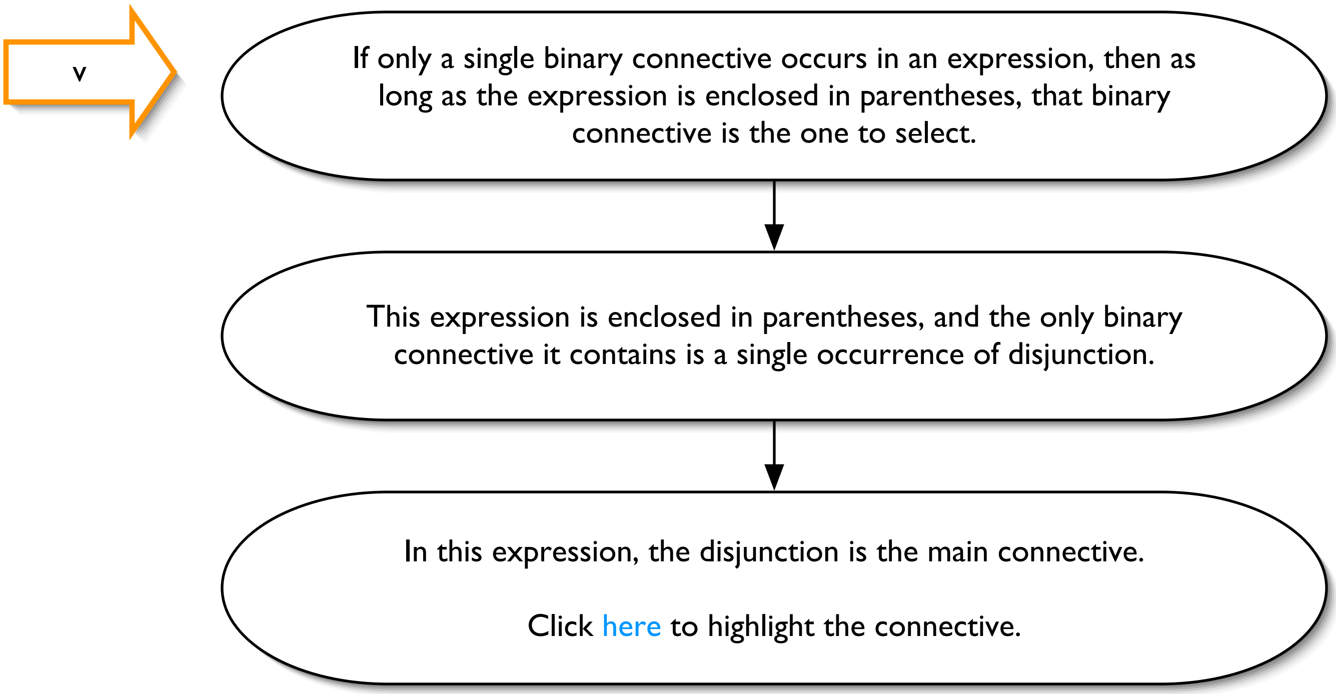


textbox

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 ((P(a) & ¬R(a,b)) → (S(b) v S(ab))), so the expression that should go at the end of this branch is the portion of that between the right outer parenthesis and the conditional.

The expression you should enter here is (S(b) v S(ab)).



v

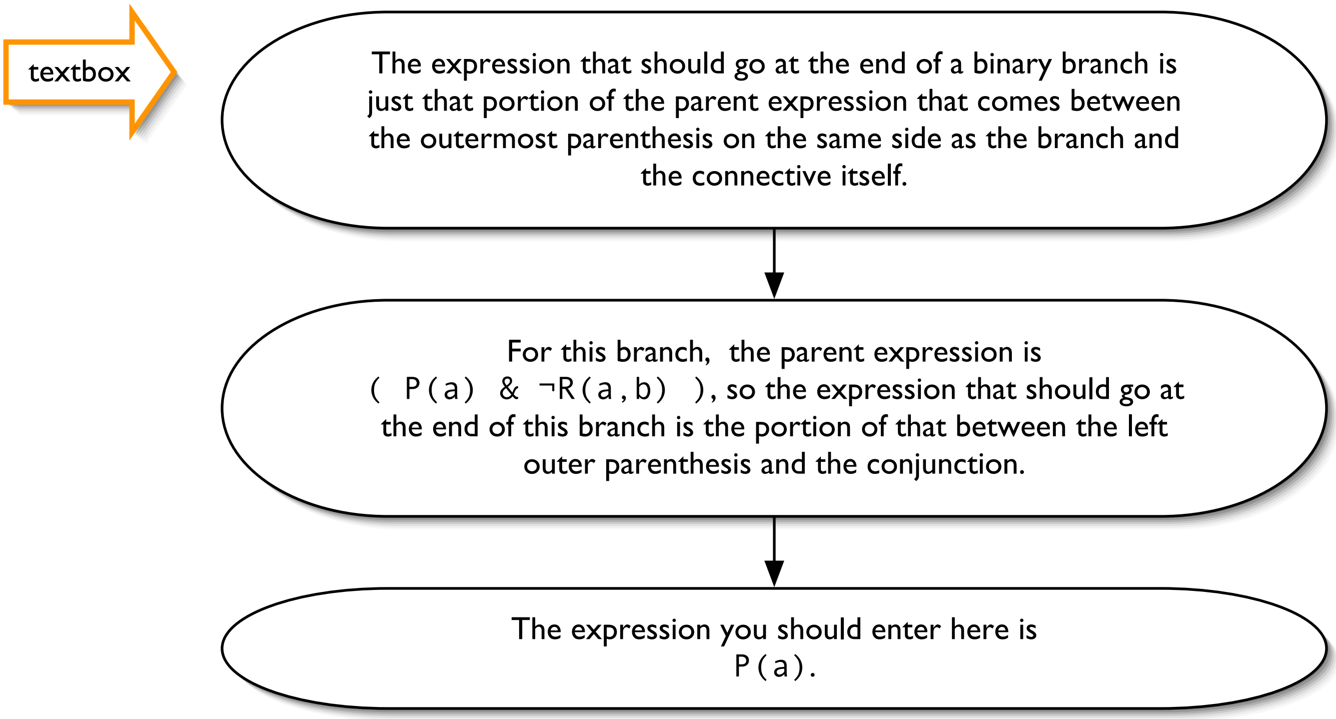
If only a single binary connective occurs in an expression, then as long as the expression is enclosed in parentheses, that binary connective is the one to select.

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

In this expression, the disjunction is the main connective.

Click here to highlight the connective.

P(a)

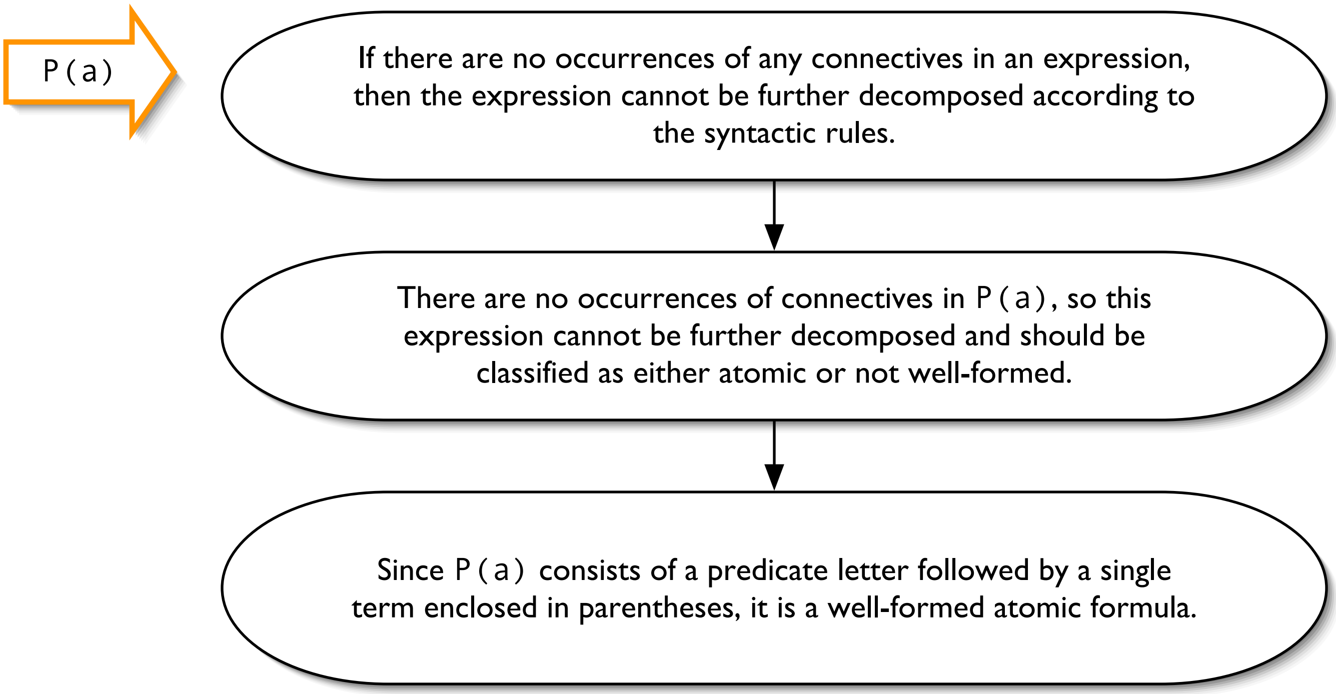


textbox

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 (P(a) & ¬R(a,b)), 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 P(a).

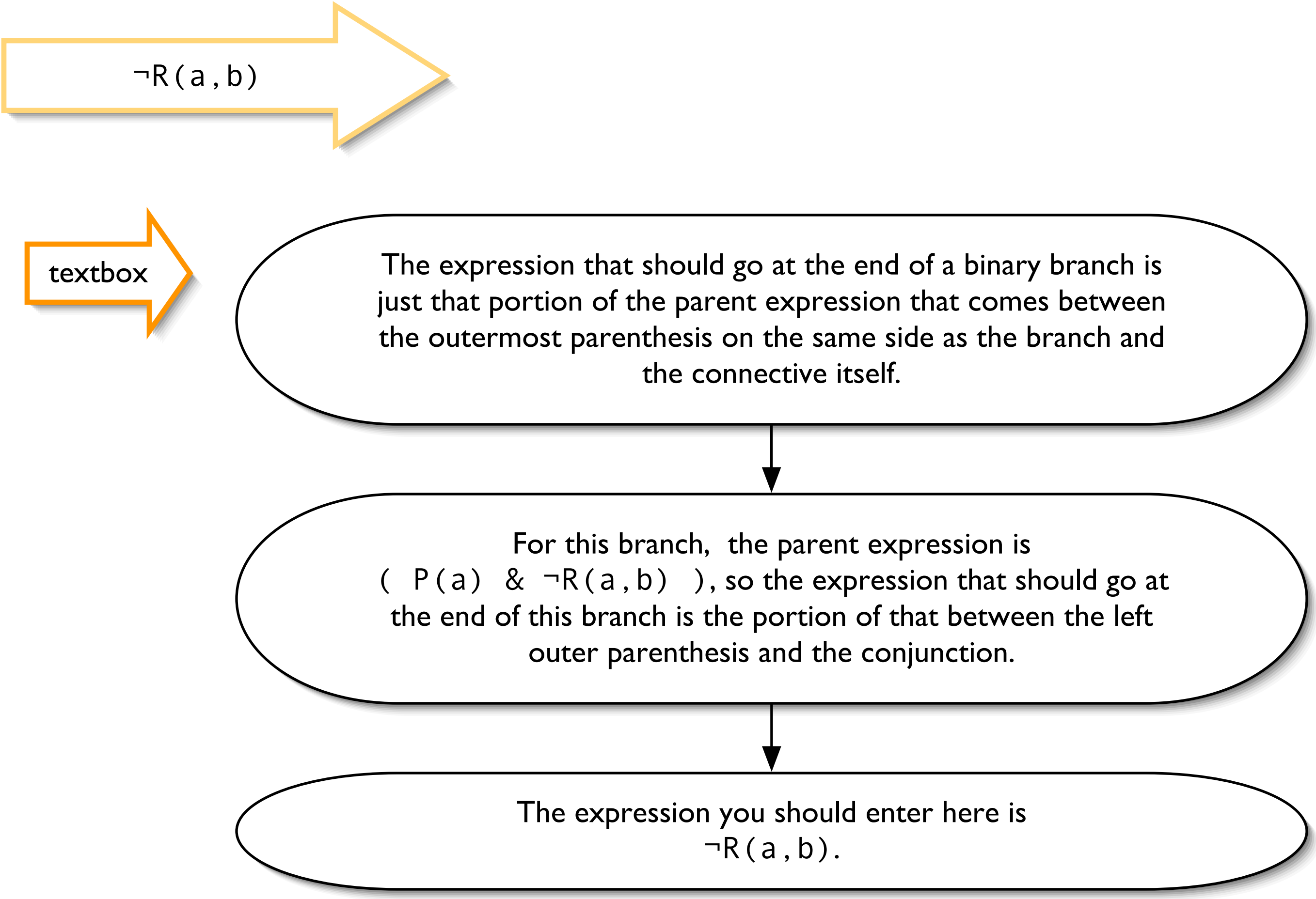


P(a)

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

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

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



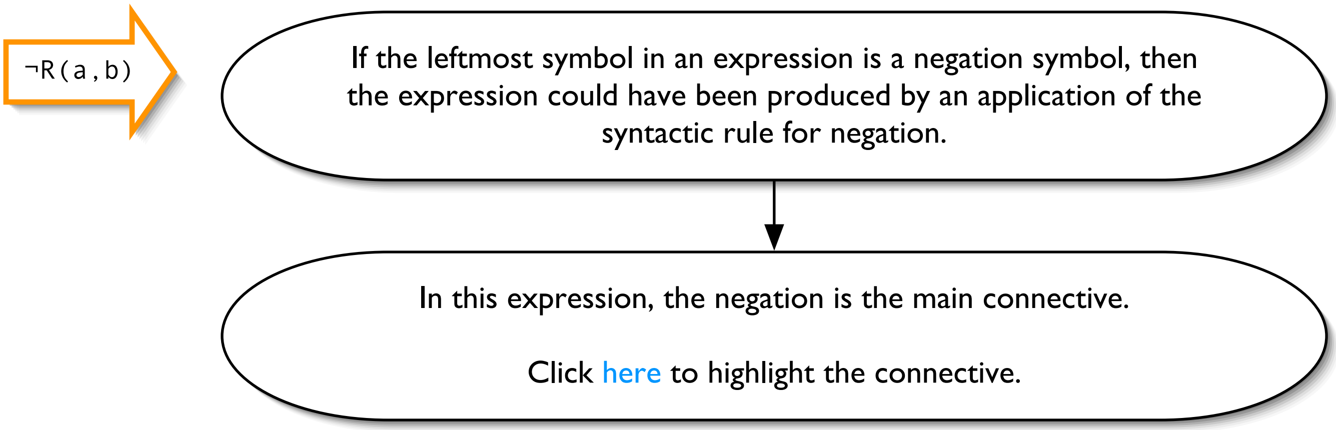
¬R(a,b)

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 (P(a) & ¬R(a,b)), 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(a,b).

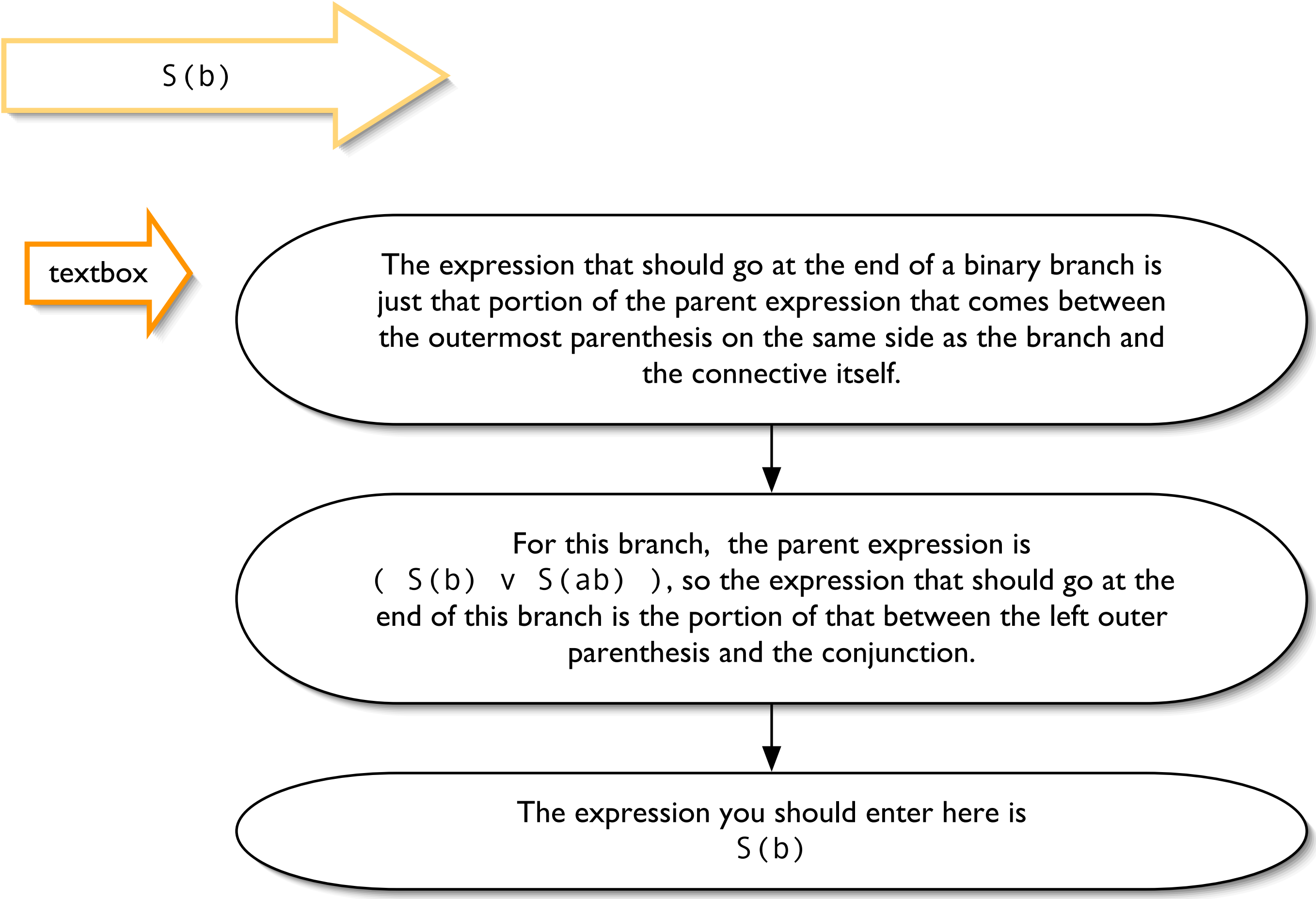


¬R(a,b)

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.



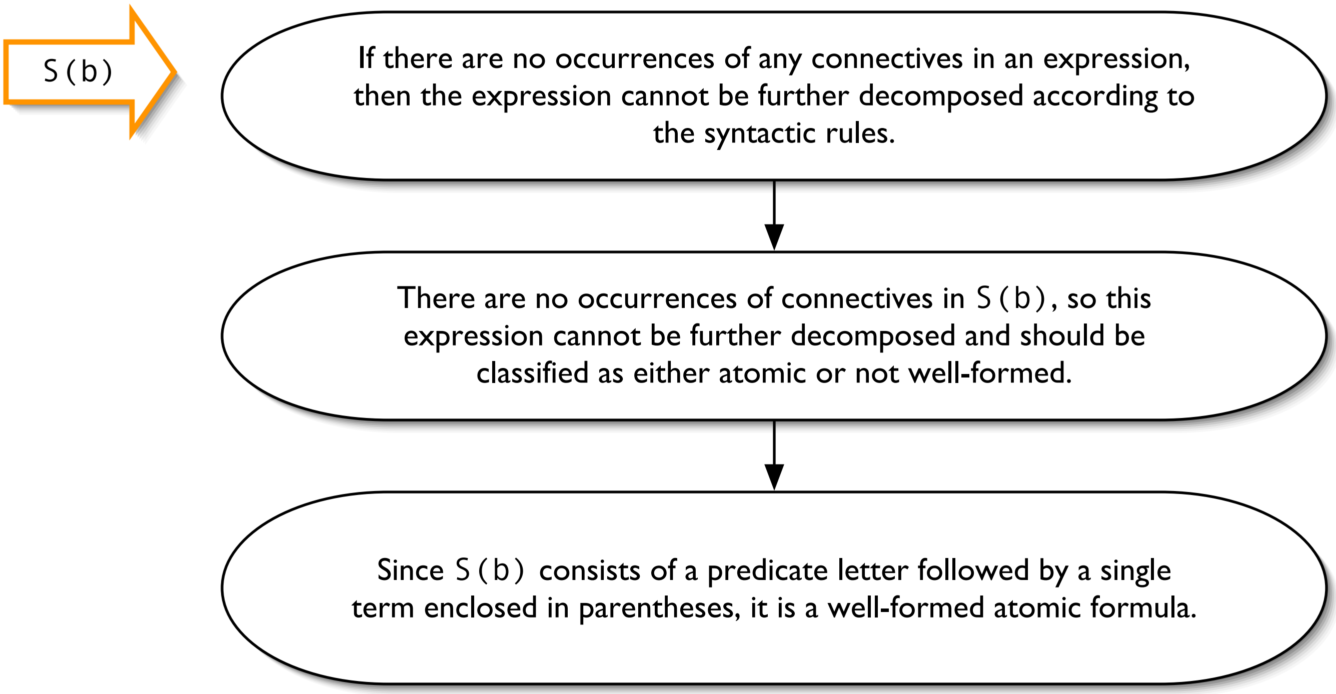
S(b)

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 `( S(b) v S(ab) )`, 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
`S(b)`

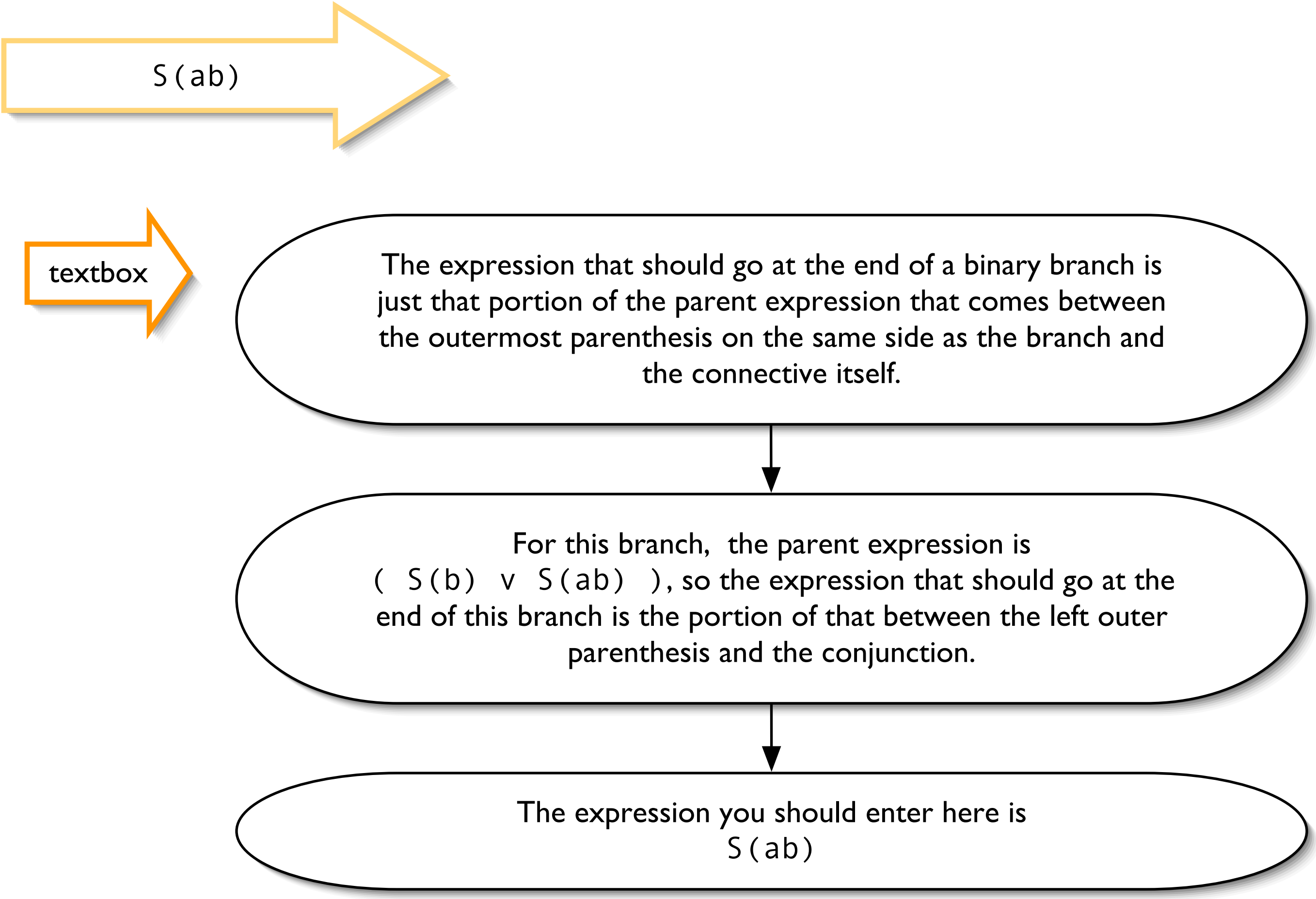


S(b)

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

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

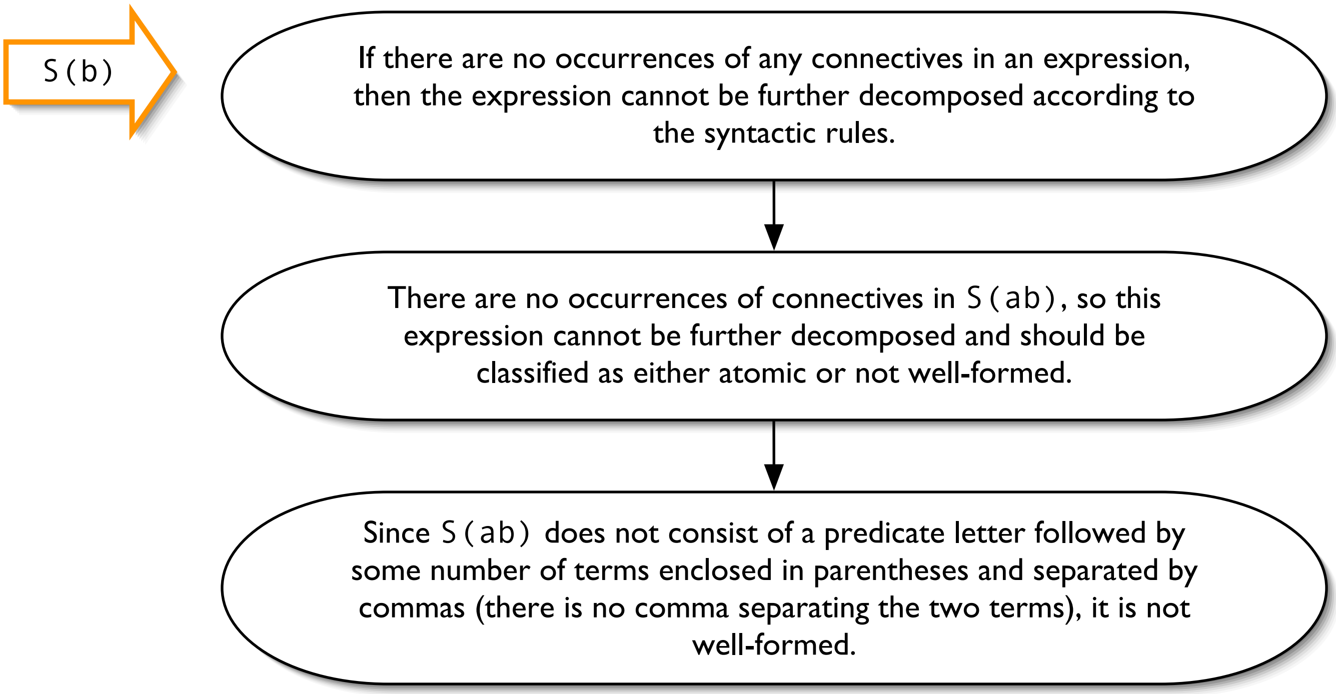
Since `S(b)` consists of a predicate letter followed by a single term enclosed in parentheses, it is a well-formed atomic formula.



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 (S(b) v S(ab)), 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 S(ab)

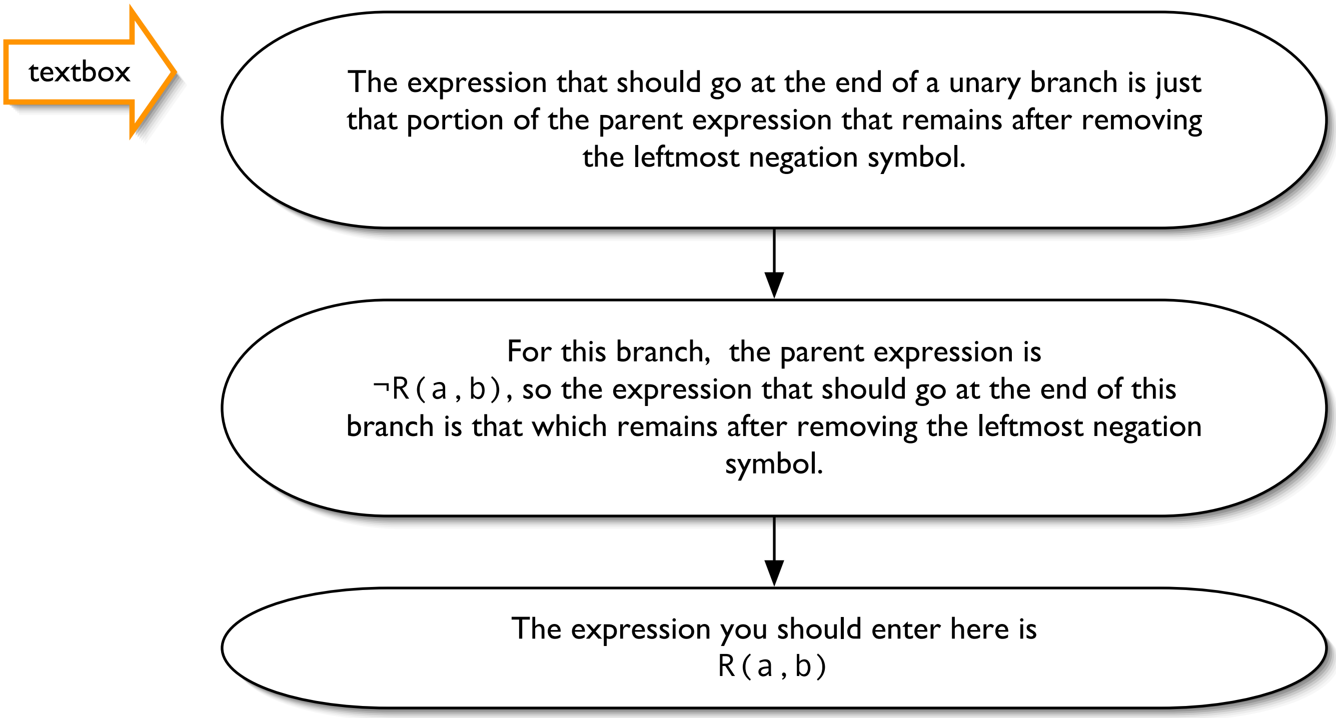


If there are no occurrences of any connectives in an expression, then the expression cannot be further decomposed according to the syntactic rules.

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

Since S(ab) does not consist of a predicate letter followed by some number of terms enclosed in parentheses and separated by commas (there is no comma separating the two terms), it is not well-formed.

R(a,b)

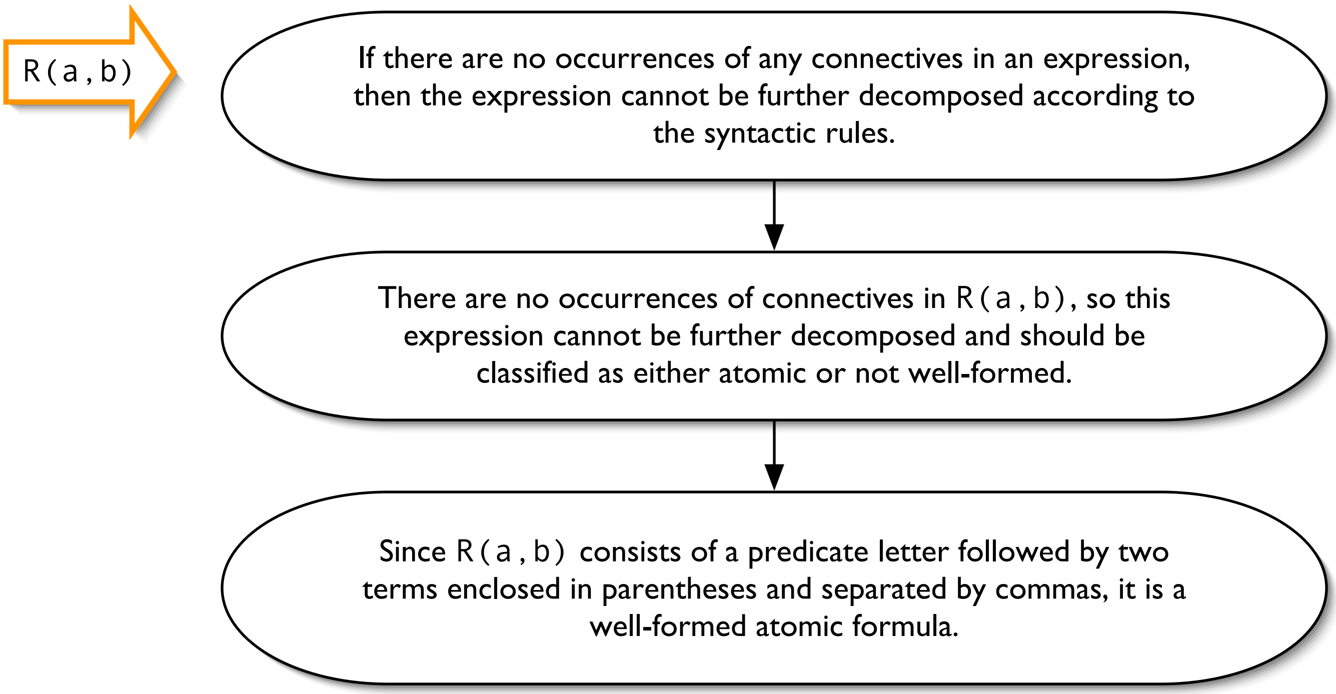


textbox

The expression that should go at the end of a unary branch is just that portion of the parent expression that remains after removing the leftmost negation symbol.

For this branch, the parent expression is ¬R(a,b), so the expression that should go at the end of this branch is that which remains after removing the leftmost negation symbol.

The expression you should enter here is R(a,b)



R(a,b)

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

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

Since R(a,b) consists of a predicate letter followed by two terms enclosed in parentheses and separated by commas, it is a well-formed atomic formula.