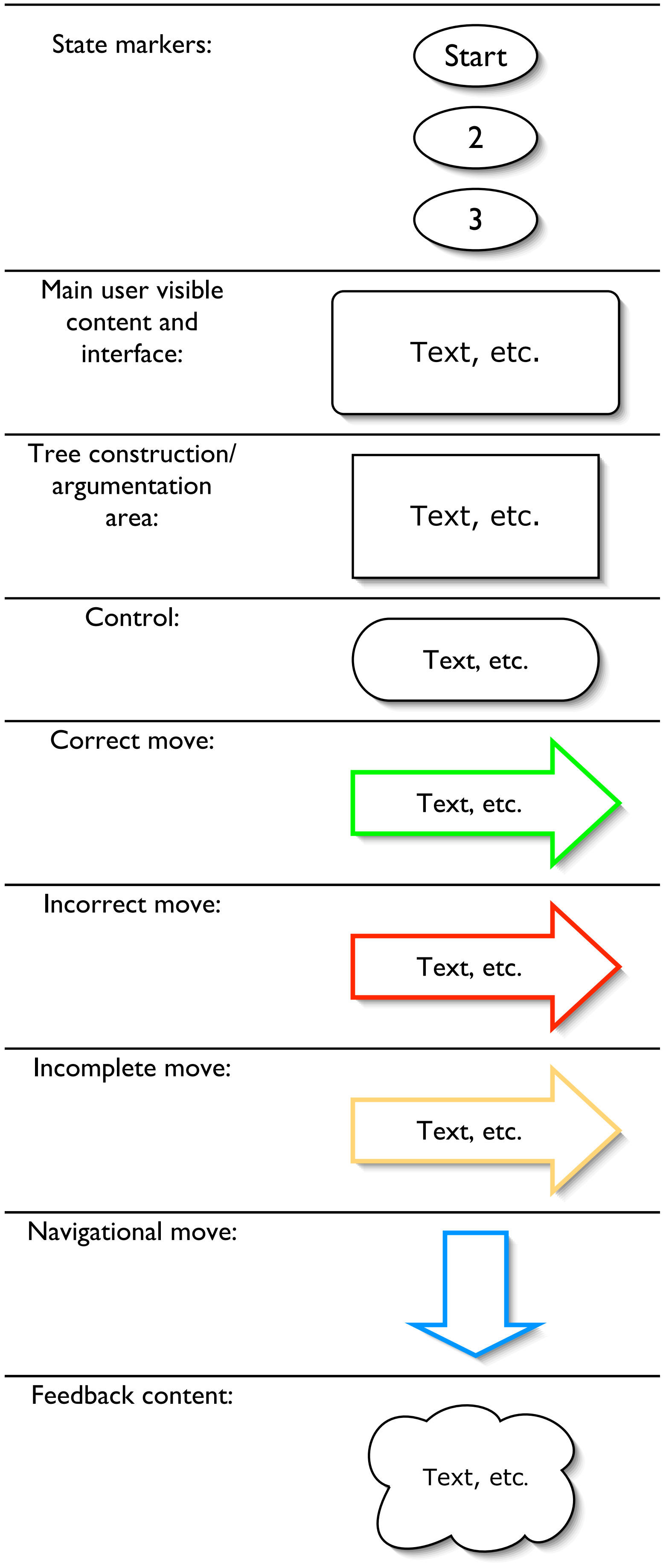
Legend:


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.

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.

(∀x)((x=a v (∃y)(¬y=a & x=y)) v (¬x=b & ¬x=c))

Create binary branch

Create unary branch

Hint

Atomic

Not Well-Formed

Create binary branch → 2

Create unary branch → 3

Atomic/ Not Well-Formed → 5

Hint → Hints

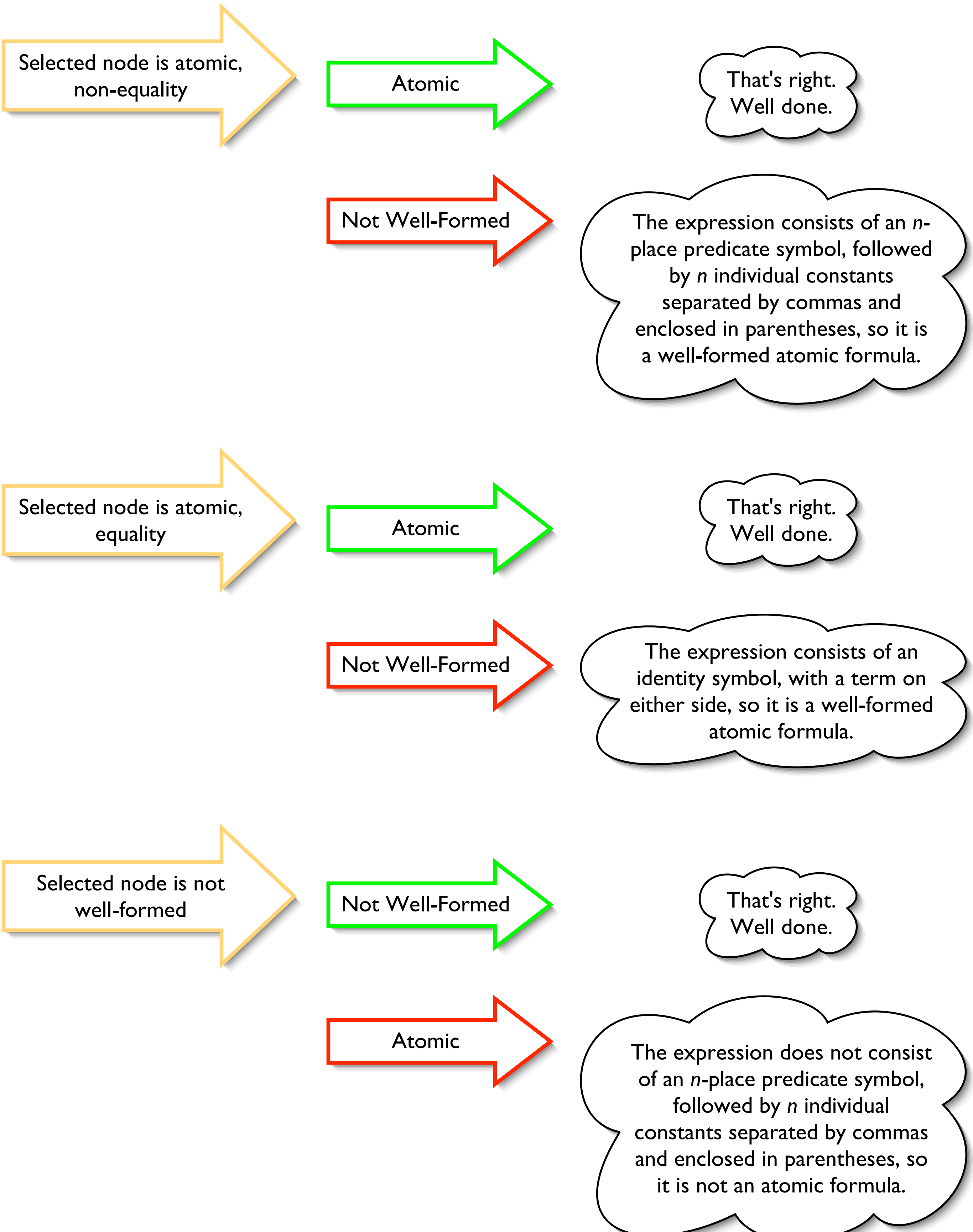
2

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, non-equality	You have selected an atomic formula.
Atomic formula selected, equality	You have selected an atomic formula. Don't forget that identity is syntactically just another two-place predicate.
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, non-equality	You have selected an atomic formula.
Atomic formula selected, equality	You have selected an atomic formula. Don't forget that identity is syntactically just another two-place predicate.
Nothing selected	[Prompt user to make a selection.]

4


Selected node is atomic, non-equality
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 atomic, equality
Atomic
That's right. Well done.
Not Well-Formed
The expression consists of an identity symbol, with a term on either side, 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:

(∀x)((x=a v (∃y)(¬y=a & x=y)) v (¬x=b & ¬x=c))

((x=a v (∃y)(¬y=a & x=y)) v (¬x=b & ¬x=c))
(x=a v (∃y)(¬y=a & x=y)) v (¬x=b & ¬x=c)
anything else C

(x=a v (∃y)(¬y=a & x=y))
((x=a v (∃y)(¬y=a & x=y))
x=a v (∃y)(¬y=a & x=y)
anything else A

(¬x=b & ¬x=c)
(¬x=b & ¬x=c))
¬x=b & ¬x=c
anything else B

x=a
(x=a
anything else A

(∃y)(¬y=a & x=y)
(∃y)(¬y=a & x=y))
anything else B

¬x=b
(¬x=b
anything else A

¬x=c
¬x=c)
anything else B

x=b
anything else D

x=c
anything else D

(¬y=a & x=y)
¬y=a & x=y
anything else C

¬y=a
anything else A

x=y
anything else B

y=a
anything else D

Feedback is on the next page.

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.

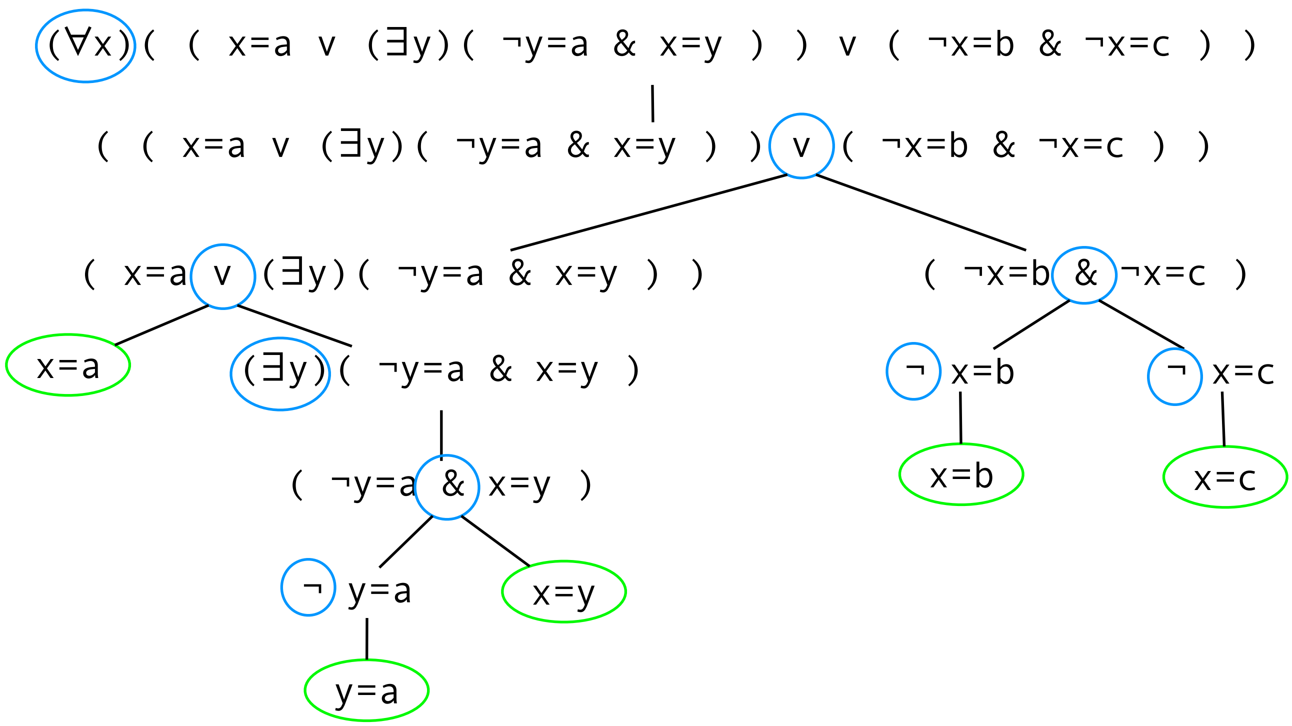
A For a binary branch, the left-hand subexpression will always consist of that portion of the original expression between the leftmost outer parenthesis and the connective that was added by the application of the syntactic rule.

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	v	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:

parsetreeconstruction6help.gif
parsetreeconstruction6hint.gif

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

(∀x) (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 universal quantifier is the main operator.

Click here to highlight the operator.

((x=a v (∃y)(¬y=a & x=y)) v (¬x=b & ¬x=c))

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
(∀x)((x=a v (∃y)(¬y=a & x=y)) v (¬x=b & ¬x=c)),
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
((x=a v (∃y)(¬y=a & x=y)) v (¬x=b & ¬x=c)).

v

If multiple connectives occur in an expression, the one with highest scope will be enclosed in only a single set of parentheses. If an occurrence has only a single unmatched parenthesis of either kind to one side, that is the occurrence to select.

In this expression, the rightmost disjunction is the main connective.

Click here to highlight the connective

(x=a v (∃y)(¬y=a & x=y))
(¬x=b & ¬x=c)

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
((x=a v (∃y)(¬y=a & x=y)) v (¬x=b & ¬x=c)), 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
(x=a v (∃y)(¬y=a & x=y))
(¬x=b & ¬x=c).

operator

If multiple connectives occur in an expression, the one with highest scope will be enclosed in only a single set of parentheses. If an occurrence has only a single unmatched parenthesis of either kind to one side, that is the occurrence to select.

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

In this expression, the disjunction is the main connective.
Click here to highlight the connective

In this expression, the disjunction is the main connective.
Click here to highlight the connective

(∃y)(¬y=a & x=y)

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 (x=a v (∃y)(¬y=a & x=y)), so the expression that should go at the end of this branch is the portion of that between the right outer parenthesis and the disjunction.

The expression you should enter here is (∃y)(¬y=a & x=y).

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

(¬y=a & x=y)
(¬x=b & ¬x=c)

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
(∃y)(¬y=a & x=y)
((x=a v (∃y)(¬y=a & x=y)) v (¬x=b & ¬x=c)),
so the expression that should go at the end of this branch is the portion of that between the right outer parenthesis and the disjunction.

The expression you should enter here is
(¬y=a & x=y)
(¬x=b & ¬x=c).

operator

If an expression contains only one occurrence of a binary connective and is enclosed in parentheses, then that binary connective is the main operator for the expression.

In this expression, the conjunction is the main connective.

Click here to highlight the connective

x=a / y=a / x=y / x=b / x=c

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 (x=a v (∃y)(¬y=a & x=y))/¬y=a/(¬y=a & x=y)/¬x=b/¬x=c, so the expression that should go at the end of this branch is the portion of that between the [left/left/right/left/right] outer parenthesis and the disjunction.

The expression you should enter here is x=a / y=a / x=y / x=b / x=c.

formula

If an expression contains no connectives and no quantifier symbols, then that expression cannot be further decomposed according to the syntactic rules, and should be classified as either atomic or not well-formed.

Since x=a / y=a / x=y / x=b / x=c contains no connectives and no quantifier symbols, and since it consists of two terms with the identity predicate between them, it is a well-formed atomic formula.

¬x=b
¬x=c
¬y=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
(¬x=b & ¬x=c)/(¬x=b & ¬x=c)/(¬y=a & x=y),
so the expression that should go at the end of this branch is the portion of that between the [left/right/left] outer parenthesis and the disjunction.

The expression you should enter here is
¬x=b
¬x=c
¬y=a.

operator

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.