

# Bertrand

A Symbolic Logic Problem Solver For The Macintosh  
Version 1.8

## CONTENTS

- 1) Contacting The Author
- 2) What Is Bertrand?
- 3) System Requirements
- 4) Paying For Your Copy Of Bertrand
- 5) Getting Started
- 6) Entering New Problems To Solve
- 7) Saving Tree, Truth Table, And Premise Files
- 8) Opening Previously Saved Files
- 9) Printing Trees and Truth Tables
- 10) Preferences
- 11) Optimization
- 12) Status Reports
- 13) The Routine Window
- 14) Tree Windows
- 15) Truth Table Windows

Appendix 1: Speed Of Problem Solving

Appendix 2: Notes On Syntax

Appendix 3: Symbol Key

Appendix 4: Tree Display Limitations

Appendix 5: The Halting Factor

Appendix 6: Update History

- 1) Contacting The Author

Please address any questions or comments to Larry A. Herzberg at [herzberg@uwosh.edu](mailto:herzberg@uwosh.edu).

## 2) What Is Bertrand?

Bertrand is a symbolic logic problem solver for the Macintosh; it is both an educational aid for students of logic and a practical tool for philosophers, logicians, computer programmers and mathematicians. Using an algorithm inspired by the "consistency tree" method found in Leblanc and Wisdom's textbook "Deductive Logic", Bertrand proceeds by decomposition and instantiation to solve first-order predicate logic statements for satisfiability, validity, equivalence, "logical truth" and "logical falsity." Subject-identity is supported. The program also produces truth-tables for truth-functional sentences.

During the solution process, Bertrand issues "status reports", giving the user insight into the method by which Bertrand constructs its tree. These status reports can be stepped through one at a time, or they can be turned off entirely to speed up the solution process. Once Bertrand has solved a given problem, the tree constructed in the process is displayed in a scrolling window. A summary of results is given in a separate window. If applicable, a model in the form of a truth-value assignment to atomic statements is provided. Tree, Truth Table, and Premise files can be saved or printed.

## 3) System Requirements

Bertrand 1.8 runs on both Power PCs and 68xxx Macs running systems 7.1 through 9.2, and has been tested on machines running OS X in Classic mode. At least 10 meg of free RAM is recommended, but Bertrand will run and solve a number of shorter problems with considerably less memory allocated to it.

4) Bertrand is freeware and open source.

Please see the license agreement for details.

## 5) Getting Started

**IMPORTANT:** You must install the Bertie font (which came with Bertrand in the original archive) before Bertrand will work.

The first time you run Bertrand, you will see an alert box asking if you have read the license agreement. If you have not, click on "Cancel" and go read it. If you have, click on "I Agree" to continue (you will never see that alert box again). You will then be presented with a Page Setup dialog box. Click on "Okay"; this will insure that the page lines displayed in tree windows provide you with an accurate representation of the page boundaries you can expect on subsequent printouts. If you click on "Cancel", you will be presented with the Page Setup dialog box the next time you run Bertrand.

## 6) Entering New Problems To Solve

Select "New..." under Bertrand's File menu. Select the kind of problem you want to solve: Satisfiability, Validity, Equivalence, Logical Truth, Logical Falsity, or Truth Table. You will then be presented with a dialog box into which you can enter the premise(s) of your problem. No statement can be longer than 255 characters. You can enter up to 256 statements in the Satisfiability and Validity dialog boxes, 2 in the Equivalence dialog, and 1 in the Logical Truth, Logical Falsity, and Truth Table dialog boxes. Click on the icons on the right side of the Satisfiability and Validity dialog boxes to scroll through premises one line or one page at a time. The Up and Down arrow keys can also be used for scrolling purposes. Copy, Cut, Paste, and Clear functions are available for use in all problem-entry dialog boxes. Just use the Edit menu or the standard keystrokes (Copy: Command-C; Cut: Command-X; Paste: Command-V).

All statements must be entered in the syntax of sentential and/or first-order predicate logic (See "Appendix 2: Notes On Syntax", below). The keystrokes needed to produce logical symbols (negation, disjunction, conjunction, conditional, biconditional, Sheffer's stroke, neither/nor, universal, and existential) can be found in the Symbol Key window under Bertrand's Help Menu (see Appendix 3 below for a complete listing of relevant keystrokes). When entering statements, these conventions must be followed:

**Predicates:** all capital letters A - Z. When not followed by a subject, these are treated as sentence letters.

**Subjects:** all lower case letters a - w; may be followed by a single subscript.

**Quantifier variables:** lower case letters x, y, and z; may be followed by a single subscript.

Subscripts: any (single) subscripted numeral  $0 - 9$  (keystrokes: option-0 through option-9).

Quantifiers: must be entered with a variable (with or without subscript) and surrounded by parentheses; e.g.  $(\forall x)$  or  $(\exists x)$ .

Identities: identity holds only between subjects. Identities like "a=b" must be notated as "=ab". Identities can close branches on a tree (i.e., produce a contradiction) in either of two ways. Example 1: "=ab" is contradicted by " $\sim$ =ab". Example 2: "Pa" is contradicted by " $\sim$ Pb" on any branch where "=ab" appears.

For more on the particular syntax of first-order logic used by Bertrand, see "Appendix 2: Notes On Syntax".

## 7) Saving Tree, Truth Table, And Premise Files

Solved problems can be saved in either of two formats. Premise files, containing only the statements that are entered into the various new problem dialog boxes, are fairly small. Tree (and Truth Table) files, which contain all of the information necessary to redisplay or print a particular tree (or truth table) without having to re-solve the problem, can be relatively large.

To save a particular Tree, Truth Table, or Premise file, bring the window containing the problem you want to save to the front by clicking on it. Then select "Save As..." under Bertrand's File Menu, and click on the kind of file you wish to save (e.g., Tree or Premises). The File Selector will then allow you to save your file to any desired destination.

## 8) Opening Previously Saved Files

Bertrand offers two ways to open previously saved problems. The first way is to select "Open..." under Bertrand's File Menu, and then use the File Selector to open the desired Tree, Truth Table, or Premise file. The second way is to double-click on the desired file(s) in the Finder; if Bertrand is already running, it will open the selected file(s); otherwise, the Finder will run Bertrand first, and then Bertrand will open the selected file(s). When you open a Premise file,

Bertrand loads the premises into the problem-entry dialog box appropriate to the problem; you can then add, subtract, or modify premises as you like before solving the problem. When you open Tree or Truth Table files, Bertrand immediately displays the information contained therein in a Tree or Truth Table window.

Note that while Bertrand can open multiple Tree or Truth Table files from the Finder, it can open only one Premise file at a time (from the Finder).

## 9) Printing Trees And Truth Tables

Before printing a tree, it is a good idea to arrange the branches so that they fit on as few pages as possible (see "Tree Windows", below). Bertrand can print trees up to three pages wide and any displayable length (see "Appendix 4: Tree Display Limitations", below). It can print truth-tables of any length and width.

To print a particular tree or truth-table, bring the window containing the problem you want to print to the front by clicking on it. If you have not yet done a Page Setup, or have changed printers since your last Page Setup, select "Page Setup..." under Bertrand's File Menu. Select options as desired. If you are printing a tree, check to make sure that all of its branches fall within the three-page-wide boundaries (represented by thick gray lines in the tree window). Initial left, right, top and bottom margins are set in Bertrand's Preferences dialog box (see "Preferences", below). The font size of the printout is the same as the font size of the window (set in the Preferences dialog). Finally, select "Print..." under Bertrand's File Menu, and set whatever options you desire.

Tree and truth-table files can also be printed from the Finder. Select the desired files and select Print from the Finder's File Menu. If Bertrand is already running, it will load the selected files and bring up the Print dialog box; otherwise the Finder will first run Bertrand, which will print the files and then quit.

## 10) Preferences

Select Preferences under Bertrand's File menu to save a variety of settings. The Preferences dialog box allows you to set the font sizes for the Tree, Tree

Info, Statement Info, Assignment, Truth Table, and Table Info windows. It also allows you to set the font size for the headers printed on every page of a print-out (which include the page number, the page orientation on trees wider than one page, and the problem result), as well as print margins (which show up in tree windows as thick gray lines). Finally, you can set Bertrand's "halting factor" here as well (see Appendix 5).

Clicking on "Okay" will reset your preferences, "Cancel" will leave them unchanged, and "Save" will save the current settings as the default settings. When you save your preferences, you also save the current Optimization and Status Reports settings (see below).

## 11) Optimization

Bertrand has three levels of optimization available. These are set under the Options menu, and can affect both the number of branches a tree will have, and the length of time it will take to construct it.

Strong Optimization checks each branch for statements of a given kind, tests each statement of that kind for how many branches it will close on decomposition (thanks to contradictions with other atomic statements on the various branches), and then finally decomposes the statement that generates the most branch closings. This can be a slow process on complex problems, but it usually generates the shortest tree.

Weak Optimization also checks each branch for statements of a given kind, but tests each statement of that kind only for whether or not it closes any branches. When a statement of a given kind is found that closes at least one branch, it is decomposed. This can be noticeably faster than Strong Optimization, but usually generates somewhat longer trees.

Optimization Off also checks each branch for statements of a given kind, and if one is found it is immediately decomposed (whether or not it closes any branches). This is usually the fastest method of solving a problem, but in some cases it will generate a tree much longer than either Optimization method.

## 12) Status Reports

Status Reports are given in the Routine Window (see below) during a tree's construction. These reports inform the user in regard to the algorithm Bertrand is using to solve the problem. Turn Status Reports off for fastest performance. Turn it on with Basic selected for next fastest, on with Detailed selected for more information but slower performance, and on with Step selected to step through the solution process one stage at a time (by hitting the Return key). Unlike Optimization options, Status Reports options can be changed "on the fly" while the solution process is occurring (by clicking on the appropriate buttons at the bottom of the Routine Window).

### 13) The Routine Window

After you click on "Okay" in a problem-entry dialog box, the Routine Window (with its green "Bertrand" title) appears with the following information: the problem type (satisfiability, validity, etc.), the optimization mode (Off, Weak, or Strong), the number of algorithm cycles Bertrand has passed through so far, the tree's current width (in terms of branch endpoints), its current length (in terms of total lines or statements), the amount of memory used (in bytes), the amount of free memory remaining in Bertrand's partition, and finally the number of seconds you've been patiently waiting for a solution -- although most problems take only a few seconds to solve, some can take several minutes, and a few can take hours or even days. At the bottom of the left side is the "Abort" button (in case your patience runs out). This button will read "Show Tree" after Bertrand solves your problem. The right side of the window is where the Status Reports (see above) are issued. On the bottom of the right side are the buttons used to control the status reports "on the fly": whether the status reports are Detailed or Basic, and also whether the Status Reports option itself is On, Off, or in Step mode.

### 14) Tree Windows

Two windows open immediately after you click on the "Show Tree" button in the Routine window: the Tree Info window and the larger Tree window.

The Tree Info window is initially open in the foreground. It gives the solution to the problem in plain English, as well as other relevant information: the length of the tree, its type (or "case") in the Leblanc and Wisdom nomenclature, the line number of the bottom statement on the branch from which the verifying truth-

value assignment (if any) has been gotten. This window can be displayed at any time by clicking on the Tree Info button at the bottom left of the Tree window.

Clicking anywhere in the Tree window brings it to the foreground (if you do not close the Tree Info window first, it will remain open in the background). Use the horizontal and vertical scroll bars to navigate around the tree.

Double-click on any statement in the Tree window (or anywhere on a statement's horizontal axis) to open its Statement Info window. Statement Info windows contain all relevant information pertaining to the statement: the problem from which it originates (in case you have several tree windows open), its length, line number, main logical operator, the index at which the main operator occurs, the statement's logical parent (from which it was decomposed or instantiated), its physical parent, its physical children, its offset from the center of the tree (in pixels), its orientation (left or right relative to the center of the tree), and finally its branch degree (i.e., how many branches stand between it and the center of the tree). Statement Info windows can be very useful in understanding long and complex trees, where a particular statement's logical and/or physical parents might be far removed from it.

A quicker way to obtain a particular statement's line number and the line number of its logical parent is simply to observe the numbers running down the left and right margins of the Tree window. The left numbers are the statements' line numbers; the right numbers are the line numbers of their logical parents. If it is not immediately apparent which numbers are aligned with which statements (which might be the case if the window is open wide), a horizontal line connecting the line number, statement, and logical parent line number can be displayed by pressing the Command key and clicking anywhere on the horizontal axis of these items.

It is often useful to rearrange the horizontal positions of the branches of a tree, which can become tangled in the process of solution. Also, although Bertrand is capable of printing trees that are three pages wide (and any length), it is occasionally necessary to drag branches that have grown out of printing range back onto the printable area. Often it is preferable to drag all branches onto the center pages, so that the print-out will be easier to read. (The printable area boundary lines are displayed as thick gray lines in the tree window; the inner page boundaries are displayed as light dotted lines).

To horizontally rearrange the branches of a tree, click on the statement you wish to move (or anywhere on its horizontal axis), and, while holding down the mouse button, drag it to its desired location; all of its physical descendants (that is, all connected statements on branches lower than it) will move with it. If the tree is in this way made broader or narrower, the window will automatically scroll in the direction of the moved statement; this reflects the fact that you have affected the total horizontal dimension of the tree. If you prefer to move a single statement without moving its physical descendants, press down and hold the Option key before clicking on the statement you wish to move.

Open branches on trees indicate a variety of things, depending on the type of problem that has been solved. The Tree Info window contains an interpretation of the existence (or non-existence) of an open branch on a tree. In all such cases, however, the presence of an open branch yields a truth-value assignment to some "atomic" statements (i.e., statements that include neither truth-functional connectives nor quantifiers) which have resulted from the decomposition and/or instantiation of the original statements. This truth-value assignment to atomic components may be interpreted as a model which verifies the solution of the problem. When there is at least one open branch on a tree, an "Assignment" button will appear next to the horizontal scroll bar at the bottom of the window. Click on this button to view a verifying truth-value assignment. If the assignment is only partial (due to the infinite nature of the model), this will be indicated in the title of the Assignment window.

A "Prune" button appears next to the Tree Info button at the bottom of all tree windows. Clicking on this button results in a "pruning" of the tree to all but one significant branch. When the solution to the problem has a verifying model, this will be the branch from which the truth-value assignment was gotten; when there is no verifying model (since all the branches close), this will instead be the longest branch of the tree.

## 15) Truth Table Windows

Truth Table windows are simpler than Tree windows. The user-entered premise appears on the top line of the window; table rows and columns can be scrolled through with the horizontal and vertical scroll bars. When the window first appears, the truth values underneath the premise's main operator (displayed in bold) are surrounded by a solid-line rectangle; the two columns of truth values

used to calculate these truth values are surrounded by dotted-line rectangles. Clicking under any logical operator will result in its truth values (and the truth values used to calculate them) being similarly boxed; this allows the user to see how each column of truth values was arrived at.

For information pertaining to the table, click on the "Table Info" button at the bottom left of the truth table window. This window will show you the problem title, problem type, total number of rows, total number of columns, premise length (in characters), number of distinct atomic statements in the premise, number of logical operators in the premise, the main logical operator, its index, and the result (which will be either "truth-functionally true," "truth-functionally false," or "truth-functionally indeterminate").

## Appendix 1: Speed Of Problem Solving

The speed at which Bertrand constructs trees is a function of many variables. First, of course, is the speed of your hardware. PowerBook owners should turn off processor recycling (see your owner's manual). Cutting down on the number of colors your monitor displays (e.g., from millions to thousands) can result in a significant performance improvement (doing this increases the speed of my G4 by more than 20%). Second is the speed of your System software. Using many extensions can significantly slow down the speed of your applications; for maximum performance, turn as many extensions off as is practical for you. Third is the nature of the problem you are trying to solve. Problems that generate trees with many branches and few contradictions can take quite a while to solve; such problems with many identities can take longer still. Fourth and fifth are Bertrand's Optimization and Status Report settings (see "Optimization" and "Status Reports" above).

## Appendix 2: Notes On Syntax

What follows is a rather quick and sloppy guide to the syntax of sentential and first-order logic notation as used by Bertrand. If you do not know the basics of first-order predicate logic, read the first few chapters of any introductory symbolic logic textbook before trying to use Bertrand.

The minimal legal statement consists of a single sentential symbol; for instance, "P" (which can be used to symbolize any sentence you like, as long as it

consistently symbolizes the same sentence throughout a given problem). Any capital letter from A through Z can be a sentential symbol. These can be negated using negation symbols " $\sim$ " or " $\neg$ " (e.g.,  $\sim P$ , which can be translated as "it is not the case that P"). Sentential symbols (and their negations) can be separated by logical operators: the conjunction ("and") symbols " $\&$ " and " $\wedge$ " (e.g.,  $P\&Q$ ), the disjunction ("or") symbol " $\vee$ " (e.g.,  $P\vee Q$ ), the conditional ("if-then") symbols " $\rightarrow$ " and " $\supset$ " (e.g.,  $P\rightarrow Q$ ), and the biconditional ("if and only if") symbols " $\leftrightarrow$ " and " $\equiv$ " (e.g.,  $P\leftrightarrow Q$ ), the Sheffer's Stroke symbol " $\mid$ " (e.g.,  $P\mid Q$ ), and the joint denial symbol " $\downarrow$ " (e.g.,  $P\downarrow Q$ ). Appendix 3 lists the keystrokes needed to access these symbols.

Capital letters function as predicate symbols when followed by a subject (any lower-case letter from a - w); e.g.,  $Pa$ , which might be used to symbolize the English subject-predicate sentence, "Alvin is a preacher." Since subject letters can be followed by a single subscript (e.g.,  $a_1$ ), you can utilize as many as 253 subjects in a single problem (subscripts are entered by holding down the Option key and hitting the desired numeral). Note that both the subject-predicate order and the symbol that is capitalized is just the opposite of English. More than one subject can follow a predicate; e.g.,  $Lca$ , which might be used to symbolize "Cathy likes Alvin." These too can be separated by a logical operator (e.g.  $Pa\rightarrow Lca$ , which might be used to symbolize "If Alvin is a preacher then Cathy likes Alvin"). You use parentheses to construct more complex sentences. For instance,  $(Pa\rightarrow Lac)\vee(Pb\&Lbc)$ . "It is not the case that Alvin is a preacher and Cathy likes Alvin" might be symbolized by  $\sim(Pa\&Lca)$ , but note that such English sentences are ambiguous; it might just as well mean the logically quite different  $\sim Pa\&Lca$ . Hence the need to use parentheses carefully.

Suppose that Cathy is really Alvin in disguise; they are actually the same person. This identity would be symbolized " $=ac$ ". Bertrand treats the " $=$ " sign as a special predicate;  $=ac$  and  $\sim =ac$  are treated as contradictories, but so are, e.g.,  $Pa$  and  $\sim Pc$  (or  $\sim Pa$  and  $Pc$ ) if they appear on a branch with  $=ac$ . The transitivity, symmetry, and reflexivity of the identity relation are recognized.

English statements that involve the terms "all" or "everything" are known as universal generalizations; those involving "some" or "something" are existential generalizations. Such "quantificational" statements are symbolized using "quantifiers". Bertrand uses the symbol " $\forall$ " to help symbolize universal generalizations, and the symbol " $\exists$ " to help symbolize existential generalizations. These symbols must be followed by variables which "stand in", so to speak, for subjects.

When you enter a quantificational statement into Bertrand, it must be done in a certain style. Quantifiers (and the variables they quantify) must be surrounded by parentheses, which must in turn be immediately followed by a complete symbolic statement; for instance,  $(\forall x)Px$ ,  $(\exists x)\sim Px$ ,  $(\forall x)(\sim Px \rightarrow Fx)$ . Variables, like subjects, can be followed by a single subscript (e.g.,  $(\forall x_1)Px_1$ ). This effectively gives you 33 possible distinct variables per statement-phrase to work with. The minimal complete sentence that includes a universal quantifier, e.g.,  $(\forall x)Px$ , might be read, "For all x, x is pink," or less awkwardly, "Everything is pink". The minimal complete sentence that includes an existential quantifier, e.g.,  $(\exists x)Px$ , might be read, "For some x, x is pink," or less awkwardly, "Something is pink," or perhaps "There is at least one pink thing."

The "scope" of a quantifier is determined by its position immediately to the left of either a simple sentence or a complex sentence, the complexity of which is determined by logical operators and parentheses. For instance, the following are syntactically proper statements (the first three are quantificational statements; the fourth is a disjunction):

$(\forall x)Px$   
 $(\exists x)(Px \& \sim Lcx)$   
 $(\forall x)[(Px \& Lax) \vee \sim (Dx \rightarrow Lax)]$   
 $(\forall x)(Px \& Lxc) \vee (\exists x)(Pb \rightarrow Lax)$

However, nested quantifiers with identical variables are not permitted. For instance, instead of  $(\forall x)[(Px \& Lxc) \vee (\exists x)(Pb \rightarrow Lax)]$ , one must make use of distinct variables and write:  $(\forall x)[(Px \& Lxc) \vee (\exists y)(Pb \rightarrow Lay)]$ .

Of course, quantified and non-quantified phrases can appear in the same statement:  $(\forall x)(Px \& Lcx) \vee (\sim Rd \leftrightarrow Df)$ . And quantifiers can themselves be negated:  $\sim(\forall x)Px$  might be used to symbolize, "It is not the case that everything is pink." Note that this is logically quite different from  $(\forall x)\sim Px$ , which might be read "Everything is not pink." The first statement asserts that there is at least one (and perhaps only one) non-pink thing; the second statement asserts that everything is non-pink. Negated universal generalizations are logically equivalent to existential statements, and negated existential statements are equivalent to universal generalizations. For instance, if  $(\exists x)Px$  is used to symbolize "Something is pink",  $\sim(\exists x)Px$  should be understood as asserting "It is not the case that something is pink," which is to say that everything is non-pink, or  $(\forall x)\sim Px$ .

### Appendix 3: Symbol Key

Bertrand uses the following keystrokes to produce symbols

Symbol	Use	Common English	Bertrand Keys
$\leftrightarrow$	Biconditional	If and only if	Option-B
$\equiv$	Biconditional	If and only if	Shift-Option-B
$\rightarrow$	Conditional	If...then...	Option-C
$\supset$	Conditional	If...then...	Shift-Option-C
$\&$	Conjunction	And	Shift-7
$\wedge$	Conjunction	And	Shift-Option-7
$\vee$	Disjunction	Or	Option-O
$\downarrow$	Joint Denial	Neither...nor...	Shift-Option-\
$\sim$	Negation	It is not the case that	Shift-`
$\neg$	Negation	It is not the case that	Option-L
$\lvert$	Sheffer's Stroke	Not both	Shift-\
$\forall$	Universal	All	Option-A
$\exists$	Existential	Some	Option-X
$_0\dots_9$	Subscripts		Option-[numeral]

Note: keystrokes may vary on non-USA keyboards; the second conjunction symbol may be mapped to Shift-Option-Y. If in doubt, check your keyboard mapping utility.

### Appendix 4: Tree Display Limitations

While Bertrand's ability to solve problems is limited only by the available amount of RAM in the computer and the level of patience in the user, the graphic display of trees is limited by the positive 16-bit screen dimensions. Due to this limitation, when Bertrand tries to draw a line between statements that are more than this value apart (in pixels), the line "wraps around" the screen. When Bertrand detects this problem, it displays only the Tree Info window and the premises of the problem, along with the suggestion that the user try a smaller font size for the tree.

## Appendix 5: The Halting Factor

Since universal statements are re-instantiated whenever new subjects appear on a branch due to the instantiation of an existential statement, when the instantiation of a universal statement directly or indirectly produces an existential statement, an algorithm loop can occur that never halts. Bertrand detects such loops, but unfortunately not all of them are nonhalting. Sometimes after several such loops, the tree will close. Sometimes a person can tell by inspecting the tree that the loop will never halt, but no computer program can; this is the "halting problem" that causes quantificational logic to be only "semi-decidable".

To deal with this problem, Bertrand halts after detecting a certain number of such loops on a given branch. The user can set this "halting factor" to any number between 1 and 256 in the Preferences dialog box. The default factor is 5. Bertrand is modest about interpreting such trees, admitting in the Tree Info window that it cannot decide whether or not the tree closes (although a human being inspecting the tree might be able to do so).

## Appendix 6: Recent Update History

Version 1.8: Bertrand now can solve problems in the background. The user can freely access other programs, and work with all other aspects of Bertrand (including saving and printing files), while it is solving a problem. Optimization levels can now be changed "on the fly". Added "Branches Closed" line in the Routine Window. Fixed bug that caused premise files saved out of opened tree files to cause a file system error when opened. Many other internal improvements. Tested on OS 9.2.2 and 10.1 (in Classic mode).

Version 1.7: Added user-settable "halting factor" and improved Bertrand's interpretation of "case 4" trees (see Appendix 5); significantly increased algorithm speed (up to 10 times on trees with many branches, especially with Optimization off); fixed bug that caused Bertrand sometimes to crash on trees with more than 256 branches. Finally, while Bertrand is still shareware, new users no longer need to obtain a registration code in order to print and save problems.

Version 1.6: Fixed bug that caused Bertrand sometimes to miss contradictory atomic statements when they were entered as premises in a satisfiability or validity problem.

Version 1.5: First version compiled by CodeWarrior. Fixed misalignment of truth table rows when printing to high-resolution printers. Fixed "Error writing resource" when Preferences repeatedly saved. Improved Bertie font (Greek characters added; better spacing). Enlarged and improved Help windows.

Version 1.4: Fixed bug that caused text to partially overwrite the "Show Tree" button on some OS 9.x machines.

Version 1.3: Fixed bug that caused Bertrand to "hang" on certain machines during searches for Existential statements.

Version 1.2: Sheffer's Stroke and Joint Denial added. 10-sessions pre-registration limit added.

Version 1.1: Bertrand now correctly recognizes identity reflexivity.