Shrub — A Tool for Visualizing Procedure Activity in Icon

Nick Kline

Department of Computer Science, The University of Arizona

Procedure Activity

In most programming languages, the sequence of procedure calls is not of much interest, since it is linear. For example, both C and Pascal have a very simple sequence of actions that procedures can perform: they can be called and call other procedures, and they can return. Procedures cannot stop somewhere internally and pause, and then be restarted later. Branching in their call trees is not possible.

In Icon, however, a call tree can have branches because of other possible types of procedure states and activities [1]. Icon procedures can suspend. A suspended procedure returns a value but maintains its state, including where it suspended and the values of its local variables. A suspended procedure subsequently can be resumed to continue its execution.

Procedure activity can be represented as a tree rooted in the initial call of the main procedure. Suspension causes the call tree to branch.

Each node has a current state, which can be one of *suspended*, *active*, *waiting*, *returned*, or *failed*. Suspension has already been discussed. A procedure is active when it is currently being executed. Many procedures spend most of their time in waiting for a called procedure to return.

In Icon, procedures can terminate by returning or failing. Returning is similar to returning in most imperative programming languages, while failure is somewhat different. Failure returns no value, but failure can be detected and used in controlling program flow. It is not possible to resume execution of a procedure that has returned or failed.

A suspended procedure activation can also be terminated without being resumed. This occurs when control backtracking points are discarded on exit from a *bounded expression*. Bounded expressions occur in specific lexical contexts, such as loop control clauses [1].

A procedure event occurs when the state of a procedure changes. The events are *called*, *returning*, *failing*, *suspending*, *resumed*, and *removed*.

Shrub

Shrub is a visualization program that displays the tree resulting from procedure activity in Icon programs. It obtains its data from an event stream [2]. Procedure events are processed and displayed in an X-window, with Shrub interacting with the user primarily through the mouse [3].

A box at each node in the tree represents the environment associated with that procedure. The current event number is in the upper left-hand corner. Arrows between nodes indicate the most recent execution paths between the procedures. The tree is drawn from top to bottom, and centered from left to right. As the tree grows and branches, new activations are added on the right and on the bottom.

Shrub keeps track of how many times each procedure has been resumed (with the first call being 1). Both this information and the names of procedures can be displayed or hidden by clicking on buttons.

Different colors are used to distinguish the different procedure states as follows:

state	*color*
called	black
returning	green
failing	red
suspended	grey
resumed	blue
removed	orange

Appendix A contains a user's manual for Shrub. Examples of displays produced by Shrub are contained in Appendix B.

Conclusions

The call tree of an Icon program shows both depth of procedure call and the ''breadth'' of procedure suspension. Programs, such as pattern matchers, in which procedures are called in mutual evaluation tend to show more breadth than programs that use procedures in more conventional ways.

There are many ways in which Shrub could be enhanced. Procedure events are the natural basis for updating the display of a call tree, but it may be hard for a user to find a desired point of execution in terms of procedure events. In some cases, one procedure or several procedures out of many may be of interest. A way of specifying procedure activity selectively would be useful. Other possible enhancements include more control over the display itself in terms of colors and layout.

Shrub also could be adapted to other uses. Expressions in Icon have the same evaluation characteristics as procedures: They can return, fail, or suspend. String scanning has similar characteristics. Both of these activities can be visualized using the same model as used by Shrub.

References

1. R. E. Griswold and M. T. Griswold, *The Icon Programming Language*, Prentice-Hall, Inc., Englewood Cliffs, NJ, second edition, 1990.
2. R. E. Griswold, *Event Monitoring in Icon*, The Univ. of Arizona Icon Project Document IPD152, 1990.
3. C. L. Jeffery, *An Experimental Windows Facility for Icon*, The Univ. of Arizona Icon Project Document IPD150, 1990.

Appendix A — A User's Manual for Shrub

Shrub is called as

shrub [–skip n] [–delay t] *file*

where *file* is an event stream. If no file is specified, Shrub reads the event stream from standard input.

The options are:

–skip *n* This option causes Shrub to skip to procedure event *n* before displaying the call tree. This may also be done during the execution of Shrub by pressing the rightmost mouse button. See the details below.

–delay *t* This option causes Shrub to update the screen with the next procedure event every *t* thousandths of a second. For example, –delay 500 causes a delay of one half second in updating the screen between events. In the absence of this option, the display is not updated until one of the two leftmost mouse buttons is pressed.

Interactive Control

Clicking on the rightmost mouse button allows the user to choose how far to jump ahead in the display. The user is given a new window and is prompted to enter how far to skip (in procedure events). Entering 0 causes no change.

The numberings and labelings that appear on the nodes of the call tree by default can be hidden by clicking on buttons that appear in the upper left corner of the display. See the examples in Appendix B.

The user can resize the screen, causing the tree to be redrawn to fill the newly sized screen.

The display can be terminated by clicking on a ‘‘quit’’ button.

Appendix B — Examples of Shrub Displays

The following figures are snapshots of Shrub displays. The program that produced these displays is a recursive-descent parser with procedures for matching nonterminal symbols. The relative ‘‘bushiness’’ of the call trees reflects suspended procedures matching sequences of nonterminal symbols.

Figure 1 shows the call tree early in program execution, while Figure 2 shows it at a time when a many nonterminal symbols are being matched. Figures 3 and 4 show corresponding call trees with the labelings and numberings suppressed to provide less cluttered displays, making it is easier to grasp the structural characteristics of the call tree.

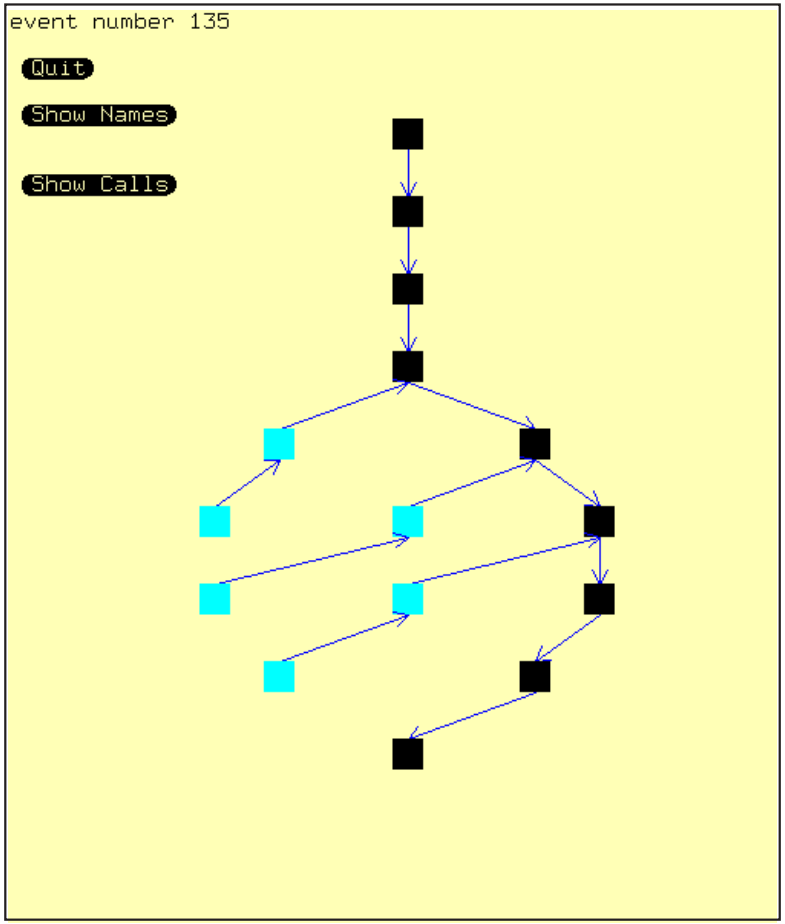

Figure 3: Early Call Tree Without Annotation

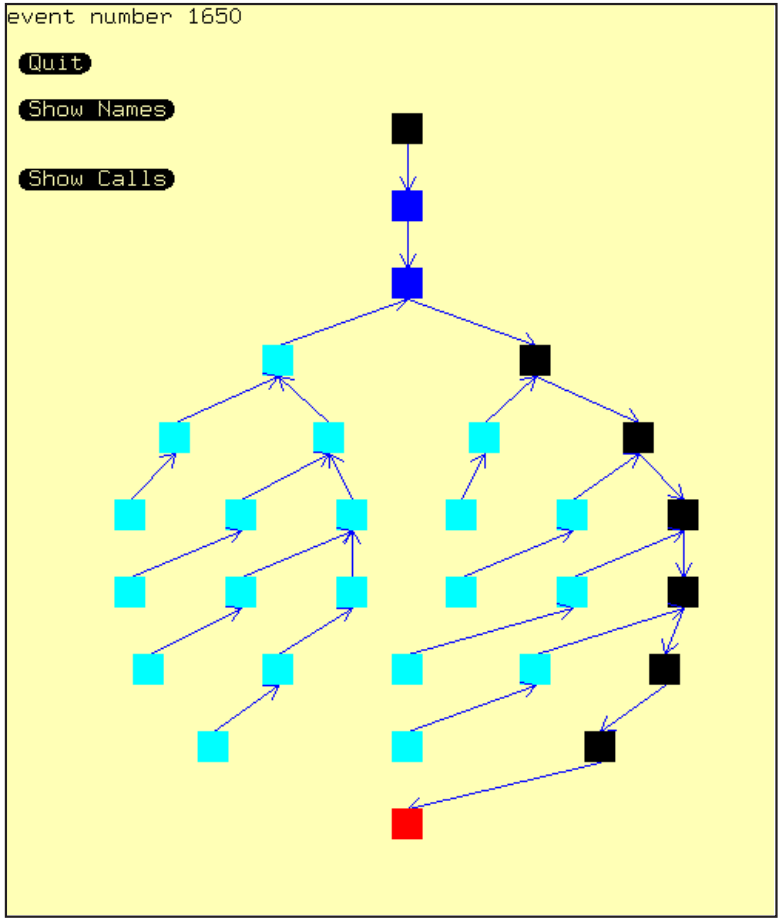

Figure 4: Later Call Tree Without Annotation

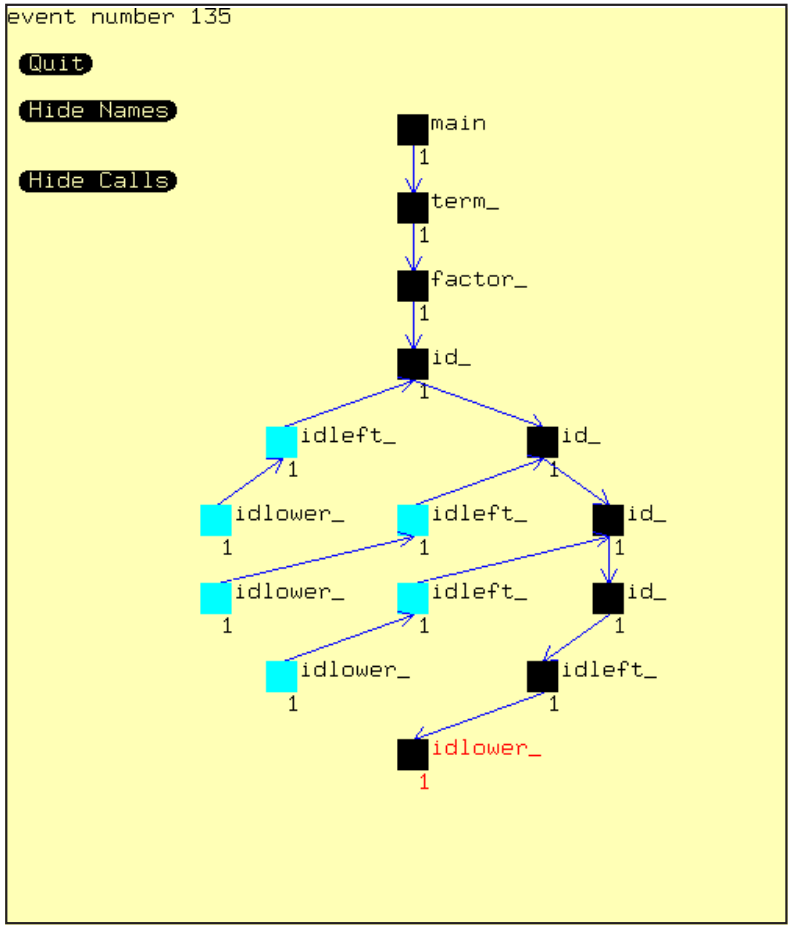

Figure 1: Call Tree at an Early Stage of Parsing

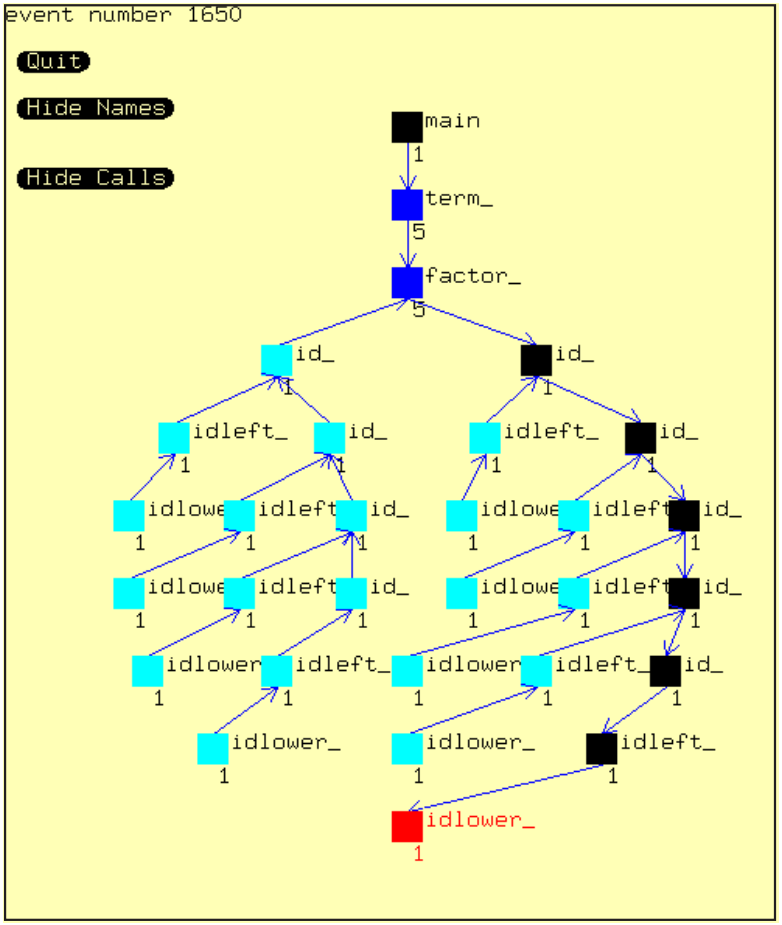

Figure 2: Call Tree at a Later Stage of Parsing