

# DDFgraph: a Tool for Dynamic Data Flow Graphs Visualization

Françoise Balmas  
Université Paris 8 (France)  
fb@ai.univ-paris8.fr

DDFgraph allows extraction and visualization of dynamic data-flow graphs computed at runtime. It currently relies on a Lisp interpreter modified to monitor data use (both of global and local variables) during program execution and to detect the corresponding data flow, namely the value of variables ‘flowing’ from one point of the program to another. A companion database stores this flow and produces, on demand, its representation as a graph for *dot*<sup>1</sup>. Furthermore, the flow graph can be manipulated, in order to reduce the size of the graph, through extraction of some parts of particular interest and/or through compaction of given subparts leaving visible only global flow.

The graphs produced by our tool support program analysis and debugging in the following ways:

- As raw graphs – i.e. without any preliminary manipulation – data flow graphs exhibit patterns showing which kinds of computation are performed. For example, the observed program (Figure 1) takes an input series and searches for a construction rule, during this particular execution in 9 different manners, appearing as ‘grapes’ hanging from the main arcs. During these searches, the program finds 2 rules permitting further extension of the series, which are recognizable in the two very distinct cyclic structures.

The drawback of this approach is that the raw data flow graph of a given program is usually too large to fit on a screen.

- To reduce the size of graphs, we adjoined graph manipulating capacities to our tool. Those are, mainly, the closing of sub-graphs corresponding to control structures or function calls – hiding flow inside subgraphs, leaving thus visible only global flow to and from the corresponding parts of the program – as well as the extraction of subparts of graphs. Both operations reduce the size of the displayed graph, allowing an engineer to examine this graph as a trace of the program: the values of variables at any point of the program are accessible, executions of function calls can be traced and given instances of function calls can be searched for.

In Figure 2, we manipulated the first graph in order to leave visible only the calls to the search and construction functions. When examining the resulting graph on a screen, one can see which are the input values of these functions, for which values a rule is found and which is the resulting extended series.

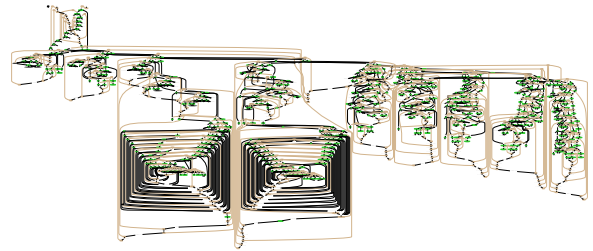


Figure 1. A raw graph with patterns

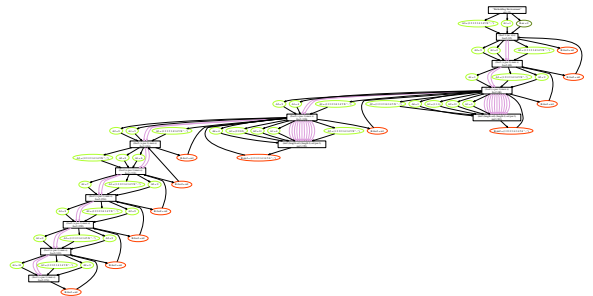


Figure 2. The same after manipulations

DDFgraph was built as a prototype to experiment with dynamic data-flow visualization. As it exhibits very promising results, we now work towards a similar tool for C programs, possibly integrated in a debugger like *gdb*, and towards the development of a query language suited to such graph manipulations.

<sup>1</sup> *dot* is a graph drawing tool from AT&T Research.