

Steps towards a GPSS block diagram system

Ingolf Ståhl*
Stockholm School of Economics

Abstract

We first note that block diagrams have been important for the learning of GPSS and have been used in most GPSS text books. The full value of block diagrams is obtained first when they can be automatically generated from program code and that block diagrams in turn can generate code. The computer generation of block diagrams requires, however, very exact rules for the block diagrams. This is lacking in the literature. We here present the first steps towards a definition of block diagram rules that can be the basis for GPSS block diagram generation, in particular for micro-GPSS. Since there are several micro-GPSS based diagram projects on its way, in different GPLs, we suggest that the main calculation work be done by the basic micro-GPSS engine, GPSS.EXE.

1 Importance of block diagrams for GPSS

GPSS has all since its first version in 1961 been using some kind of block diagram symbols. Although the symbols used and the way the block symbols have been connected have changed from book to book, several symbols have remained the same over the years. For example, 13 block diagram symbols used in a GPSS II manual from 1963 [IBM63] are the same as those used in the present version of WebGPSS.

Most GPSS books use some sort of block diagram. Examples of GPSS textbooks with many GPSS block diagrams are [Sch74], [Sch91], [Stå90], [BCS89], [BKP76], [Gor72], [Gor78], [ThT92], [WaB89] and [HoP90]. Schriber's Red book [Sch74] has more block diagrams than any other GPSS books, in fact over 60, and has set the standard. There are also some general simulation text books that present at least one block diagram in connection with their short presentation of GPSS, such as, for example, [BaC84], [LaK91] and [Ste94]. It should, however, be noted that there are also some GPSS text books that present many programs in text, but contain no block diagrams at all. Examples are [FrL79], [Chi92], [Sol83] and [ODo79].

In my opinion, the GPSS block diagram is, however, one of the major competitive advantages of GPSS. It has many strong competitors when it comes to text based simulation languages, but the GPSS block diagrams have unique features. I have in my teaching of GPSS, starting with GPSS/360 back in the 70s, found block diagrams very useful for the students. A proper block diagram conveys the logic of the simulation model better than the corresponding text code. I believe that block diagrams are particularly useful for students who are not used to programming in a text based general purpose language. This has, not the least, been the case with my business students in Sweden.

* P.O. Box 6501, SE-113 83 Stockholm, Sweden. Email: ingolf.stahl@hhs.se

2 Automatic generation of GPSS block diagrams

All the block diagrams in the books mentioned above have been drawn by hand. The full advantage of the GPSS block diagram system can, however, be obtained first when the block diagrams are generated automatically from the program text, and, even more importantly, the program text is generated from a block diagram that the user builds using a mouse. The earliest attempts of automatically generating a block diagram from the text are probably that of GPSS/PC from the early 80's. These block diagrams lack, however, connections between the blocks, in the form of paths to be discussed below. More ambitious attempts or plans were those of GRAMOS-GPSS, GPSS/VI and F. Breitenacker of the early 90's [Die93], [Bal92] and [Bre92]. None of these systems are, however, available to-day.

The major effort as regards computer block diagrams has been those based on the micro-GPSS language [Stå90]. One reason for this might be that, since micro-GPSS has less than half as many blocks types as other GPSS systems, the work on such a block based system is a lot easier. Already in the early 90s, I developed GPSSDIA [Stå93], which on the basis of a text program could generate the block diagram. It could, however not work the other way around, but many of my students requested a system where one with the block diagram as a basis could build up the GPSS program in the form of code text. In fact, one of my micro-GPSS students already in 1992 started on such a software, called GPSSEDIT [Nyw92]. This was built in Turbo Pascal for Windows. Although the system had several promising features, it was lacking in several aspects and it was never fully completed. I had then also started work on my own version of such a system, based on GPSSDIA. This system, GPSSGUI, was also developed in Turbo Pascal, but for DOS.

The idea behind these systems is that the user by choosing different block symbols from a block symbol menu can first build up the skeleton of the program in the form of the block diagram. By next clicking on a block in this block diagram, a dialog is opened and the user can input the operands of this particular block.

The work on such a GUI system did not take off until 1997, when we got financing from a Swedish educational foundation to develop a Web-based GUI of this type [HeS99]. This system, called WebGPSS, which was developed in Java, became available on the web in 1999. Parallel to this, a similar, Windows-based system, WinGPSS, also using micro-GPSS, was developed in Magdeburg, by H. Herper, A. Krueger and H. Schlifke, using Delphi [HeS99]. WinGPSS has more recently included a simple way of animating the program by having simple tokens, representing the transactions, move through the block diagram, thus animating each simulated event. I have also presented a sketch of such an animation system for micro-GPSS, using Proof [Stå00]. This involves the automatic definition of paths for the Proof layout files.

It should be mentioned that due to various problems with Java applets on the web [Stå05], we have also developed a web-independent version of WebGPSS, available on a CD. Problems with Java have also caused M. Lindhe to start sketching on a micro-GPSS-based GUI for C#. Remaking WebGPSS using Delphi has also been contemplated. It should finally be mentioned that that we have on the Web recently found a block diagram system based on micro-GPSS, called Simulaworks, developed in Egypt.

We can hence note that there are a handful of efforts towards constructing a GUI system, with symbol menus and block diagrams, based on micro-GPSS. This implies that

in order not to have a lot of duplicated efforts, with several developers writing code for the construction of the block diagram in different languages, such as Java, Delphi, C#, etc., most of the calculations needed for the construction of this block diagram should be done by one single set of algorithms, mainly to be used by the micro-GPSS engine, GPSS.EXE.

3 Issues in the generation of block diagrams

We shall hence devote the rest of the paper to a preliminary discussion of some general principles for the automatic generation of these block diagrams. It should first be noted that the principles for GPSS block diagrams have not been discussed at any length in the literature. The reason for this is most probably that all block diagrams in the literature were hand-made, in many cases in the final stages drawn by people at the publisher with little knowledge of GPSS and who do not always follow the rough sketches made by the author. In the case of the automatic generation of block diagrams very exact rules are, however, needed. Here we are not in a position to provide the final exact set of rules, but more like first steps for this. It seems important to start a discussion to get some feed-back before work gets down to the very precise details.

It should be noted that the main influence for the block diagrams as provided by the systems mentioned in section 2 above has been Schriber's Red book [Sch74], which no doubt has also influenced the block diagrams in most later books. It seems, however, not possible to distill all the principles from this book. The automatic generation of block diagrams to be seen mainly on a computer screen is a process quite different from manually drawing block diagrams to be seen in a book. Starting to think through the issues in more detail, I think that it appears clear that the diagrams based on the new principles will in some details have to differ not only from the diagrams of Schriber, but also from those of the present systems, such as WebGPSS and WinGPSS.

It should in this connection also be noted that most of these principles, regarding e.g. the location of the blocks and the paths, could also apply to other GPSS systems, such as GPSS/H and GPSS World.

We shall hence next turn to the question of what characteristics are suitable for a block diagram system for GPSS, in particular micro-GPSS. There are a number of significant issues involved.

3.1 Free or fixed form format?

The first issue deals with the way the block diagram is built up. Should it only be possible to place the blocks in some fixed places or should it be possible to place a block anywhere, completely freely, on the workspace. One cannot from the hand-made drawings in the text-books determine exactly what positioning is allowed, but for computer generated drawings these principles **must** be determined. While most other systems, like e.g. Arena, allow a block to be placed anywhere on the work area, the main four micro-GPSS based systems mentioned above (WebGPSS, WinGPSS, GPSSEDIT and GPSSGUI) have all been based on the principle that a block can only be placed in a limited number of positions. While the free positioning of a block requires drag-and-drop, the positioning in a fixed position scheme can be done by point-and-click. The fixed positioning systems generally automatically choose the place where the next block shall

be placed and a new symbol is then placed on this spot after the point-and-clicking on this block symbol in the symbol menu.

It should in this connection be mentioned that one important difference between WebGPSS and WinGPSS is that WebGPSS only allows for point-and-click, while WinGPSS as its first choice has drag-and-drop placement of the blocks in the block diagram. The form of the block diagram is, however, also for WinGPSS independent of how the symbols are brought from the symbol menu to the block diagram. Thus the slower drag-and-drop method really does not give any advantage in the case when the placement of the block symbols is fixed.

The main principle for the fixed positioning system is that each block in the block diagram should have an exact place in terms of a column and a row of the block diagram. While the width of each column does not have to be of exactly the same size, each row should have the same height. The width of each column will be determined by the number of connecting paths and the length of certain operand texts, as discussed below. We can hence see the block diagram as consisting of a number of cells, where the cells should all have the same height. All cells in the same column should have the same width and be in a straight line under each other, but cells in different columns could have different widths. I have experienced the advantage of such a fixed positioning and point-and-click system when watching my students construct a block diagram model very rapidly by just clicking on a number of block symbols in the symbol menu.

3.2 Connecting paths or not?

Another important issue is whether one should have connecting paths between some blocks. The paths are lines that connect a source block with one or several target blocks, i.e. blocks with addresses used in these source blocks. There are three types of source blocks, namely GOTO, IF and SPLIT blocks. We can also call these three types of blocks for jump blocks and we shall call the block addresses that are used both as operands in some jump block and as target addresses for matched addresses. Besides these matched addresses, there can be blocks with addresses that are used for other purposes, like determination of N\$ and W\$ SNAs or as pure comments for documentation. These addresses with other purposes are disregarded when establishing the paths.

The question is hence here how one displays the connection between a jump block and a matching target address. One way is to **try** to use some connecting path. The alternative would be to **only** have the jump block lead to an address symbol, such as a circle, and have the target address related to this address in the jump block have a corresponding symbol, both with the address name, a number or a letter inside the symbol. This system without any connecting lines is used by WinGPSS. I have, however, already since my start with GPSSGUI decided to use paths. The use of paths makes it much clearer which blocks are connected. This is especially true in the case of several pairs of jump and target blocks in a large block diagram. It is then very easy to see which blocks are connected to which blocks. The two block diagram on the next page might provide some idea of the benefit of using paths. The use of paths is furthermore, as noted, also essential for animation, e.g. using Proof, so that one can have the transactions move continuously from one block to another, without large jumps, which would be necessary in case one only used connecting circles.

