

Large State Space Visualization*

Jan Friso Groote and Frank van Ham

Department of Computer Science, Technische Universiteit Eindhoven,
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
{jfg,fvham}@win.tue.nl

Abstract. Insight in the global structure of a state space is of great help in the analysis of the underlying process. We present a tool to visualize the structure of very large state spaces. It uses a clustering method to obtain a simplified representation, which is used as a backbone for the display of the entire state space. With this tool we are able to answer questions about the global structure of a state space that cannot easily be answered by conventional methods. We show this by presenting a number of visualizations of real-world protocols.

1 Introduction

In the last decade, advances in computer hard- and software have made it possible to effectively analyze the behavior of complex software systems by means of state transition systems. Techniques based on explicit state enumeration can now deal with systems consisting of billions of states and have reached a scale at which they become sufficiently effective to assist in the design and testing of real world software systems. However, they also confront us with state transition graphs of enormous dimensions, of which the internal structure is generally a mystery. The most common approach to obtain insight in the structure of large state spaces is by abstracting from actions and/or state information, and by reducing the state space modulo a suitable equivalence. Although this approach maintains the behavioral aspect of state spaces, it destroys their structure. Typical questions that cannot be answered by using such techniques are:

- How many states are in each phase of a protocol?
- How many independent parts does the state space have, i.e. parts without a path between them? This question might be relevant to determine the effectiveness of different testing approaches.
- Are there hot-spots, i.e. groups of states that are visited relatively often when randomly traversing the state space? Are there parts of the state space that have extremely low probability to be visited?

In this paper we present a number of applications of a tool that can effectively display state transition systems consisting of millions of states and clarify their

* This work is a condensed version of [4] and was supported by the Netherlands Organisation for Scientific Research (NWO) under grant 612.000.101.

structure in this way. Essentially, the scalability of the technique is limited by the capacities of the graphics hardware. With the fast development of graphics hardware, our visualization technique will soon be suitable for even larger state spaces. Besides being scalable, this technique is also computationally inexpensive and very predictable, so local changes in structure do usually not lead to global changes in visualization. A prototype version of our tool is freely available for download from [6].

The basic idea underlying the visualization technique is to use a simplified representation in the form of a tree as a backbone for the entire structure. First the state space is layered using the distance of each node to the root. Then, the tree structure is obtained by clustering sets of states in each layer, such that for each set a unique path to the root is obtained. Each set of states is subsequently modeled using a disk shape in a three dimensional space. Finally, all disks are connected in a manner resembling cone trees [8], forming the shapes such as the ones in figures 1, 2 and 3. For a detailed description of the entire layout algorithm we refer to [5]. Note that visual cues such as interactive motion, colors, lighting and transparency all add strongly to the three dimensional perception of the shapes. The black-and-white still pictures in this print are by no means comparable to the onscreen images.

After visualizing the state space as a 3D-object we can use coloring to stress particular aspects of the state space. Typically, coloring can be induced by intrinsic properties such as the value of the transition label or state vector, or derived properties, such as the probability to visit a state during a random walk. The next sections discuss some interesting observations that can be made when applying this tool to a number of real-world examples.

2 The Alternating Bit and PAR Protocols

The Alternating Bit Protocol (ABP) is a communication protocol concerned with data transmission over an unreliable channel [1]. In this section we present visualizations of the ABP and one of its variations, the Positive Acknowledgement with Retransmission (PAR) Protocol that were modelled in [7]. Fig. 1 shows different visualizations of models of both protocols using two different data elements in the transmission. Individual states are depicted as spheres, with the initial state at the top, transitions between states are depicted as lines.

All visualizations depicted in Fig. 1 show a high degree of symmetry in the vertical axis, since the behavior for both data elements is identical. Note that the left visualization consists of more states than the middle one, even though they both describe the exact same protocol. This is due to the fact that the software used to extract the state space from its formal description suffers from state duplication. This is especially obvious in the bottom part of the left visualization, which duplicates itself. This can be remedied by applying a strong bisimulation reduction to the left state space, which gives us the middle state space. This visualization clearly displays regularities in the alternating bit protocol. The behavior in the top half is identical to the behavior in the bottom part, with

the distinguishing factor being the control bit sent with the data. The four protrusions on both sides of the visualization represent errors in the transmission of the datum (1^{st} and 3^{rd} from the top) or errors in the acknowledgement of a transmission (2^{nd} and 4^{th} from the top).

The PAR protocol resembles the ABP, but differs in the way errors are handled. Where ABP assumes non-lossy channels, PAR does not. To cope with this additional problem a timer is integrated in the protocol, which generates a time-out on loss or corruption of a message. This explains why the 1^{st} and 3^{rd} protrusions are much smaller, as PAR deals with errors in transmission more efficiently than ABP: On loss or corruption a timeout is delivered directly to the sender, which can then immediately resend. The ABP has to use the acknowledgement channel to communicate this.

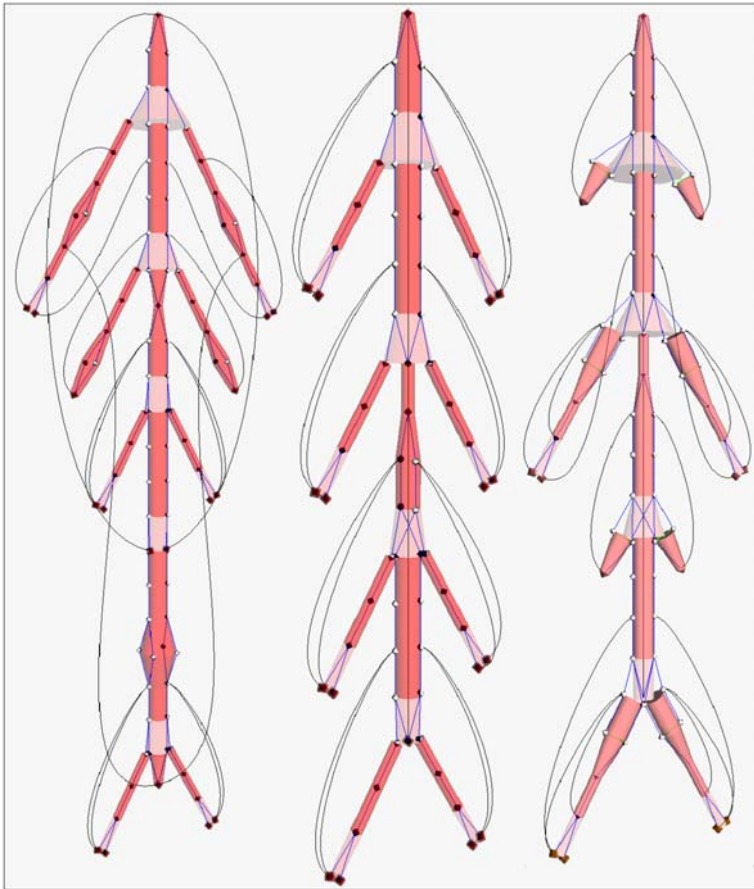


Fig. 1. Communication Protocols: ABP (*left*), ABP reduced modulo strong bisimulation (*middle*) and PAR reduced modulo strong bisimulation (*right*)

3 A Modular Jack System

This section deals with the communication protocol for a modular jack system. The protocol regulates the communication between an expandable set of industrial jack platforms, allowing the entire set of jacks to be operated from the controls of any one jack. All jacks are communicating via a ring-shaped shared bus. The protocol was formally modeled and analyzed in [3] and the exact formulation of the protocol that we visualize is distributed with the μ CRL toolset [2]. Fig. 2 shows the visualization of the protocol when it has to synchronize two separate jack platforms. One of the immediate features is the existence of two separate but symmetric legs. Although this feature is very apparent from the picture, the researchers involved only realized this when observing the picture. Generally, the protocol for k jacks has k independent legs.

Another feature that is hard to extract using conventional analysis is the fact that the protocol clearly starts with an initialization phase, the end of which is marked by a large number of returning backpointers (see arrows - Fig. 2). Looking at the formal description the initialization phase assigns a consecutive global numbering to all jacks, with each jack having equal opportunity to become the first. This also explains why the two sections are symmetrical, after all the observable behavior of the entire setup should be insensitive to the numbering scheme used. Another interesting observation is that the general behavior of a

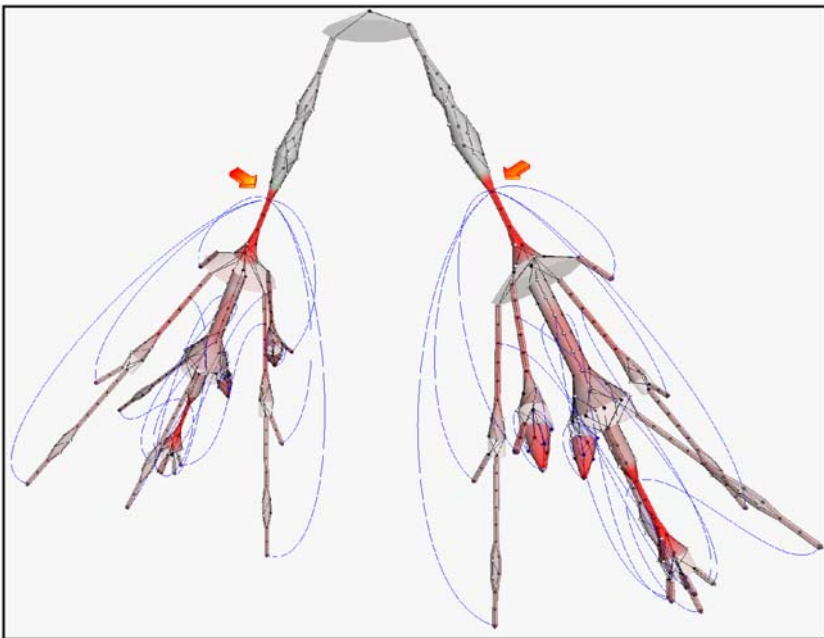


Fig. 2. Modular jack communication protocol for 2 jacks

system of two platforms is visually very similar to the behavior of a system of three platforms, consisting of approximately 3,000 states (Fig. 3). Notice the three identical sections and how different subsections of the graph visually correspond to sections of the two-platform graph. The fact that similar graphs give similar pictures is a major advantage of this method, allowing the application of insight gained in simplified versions of behavior to more complex behaviors. This visualization also shows a bug in the protocol: the small encircled section that splits off during the initialization phase of the protocol has no returning edges, which means that states at the end of this appendix are deadlock states. This was not the case in the two-jack version. Although these features can also easily be spotted by conventional analysis (actually, in this case they have been, see [3]), a picture in which you can actually point out the bug is a great asset. An additional feature is the ability to perform stochastic analysis on different

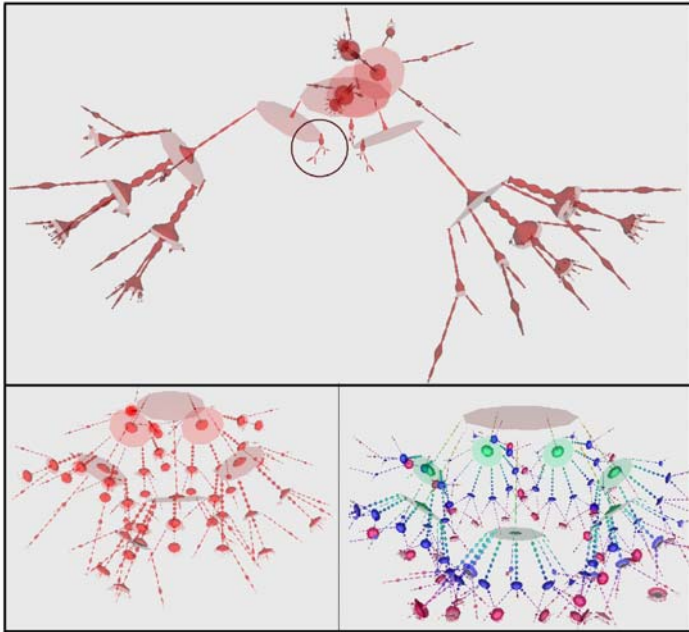


Fig. 3. Protocols for a setup of 3 jacks (2,860 states), 5 jacks (70,926) states and 7 jacks (1,025,844 states) respectively. Backpointers are hidden

automata. Assuming each transition has an equal probability of occurring, we can determine which states in the process have a relatively high probability of being visited by simulating a random walk. We can then visualize this information by coloring clusters based on these probabilities as is done in Fig 2. In this case high probability (darker) sections are located directly behind incoming backpointers and in areas with a large number of short cycles. These types of

pictures are even more useful if measured real-world probability data is used to distinct between states that are in parts of the automaton that deal with, for example, error handling and states that are part of the main body of the process.

4 Conclusions

We presented a visualization tool to gain more understanding of large finite state spaces. Its main advantages over existing contemporary visualization tools are:

- **High scalability:** by not displaying each individual state and transition we can effectively visualize a number of nodes that is at least two orders of magnitude larger than the best conventional techniques.
- **Predictability:** In contrast to some other popular graph layout approaches (such as force directed methods) this method is highly predictable. As a consequence, similar input graphs also lead to similar looking visualizations.
- **Speed:** The performance of the method is currently limited by available memory and the speed of the graphics card. The algorithm that generates the global layouts has a time complexity that is linear in both the number of states and the number of edges.
- **Interactivity:** Since it is impossible to display all information related to a state transition graph in a single picture, interaction is critical. In our tool the user is enabled to zoom into subsections of interest, highlight parts based on state vector values or transition label values and much more.

It is needless to say that we are enthusiastic about the techniques we offer in this tool. Actually, by looking at and interacting with the visualization tool we became aware of many properties of state spaces that, although sometimes obvious and sometimes more obscure, we had not realized until we saw them.

References

1. J. Baeten and P. Weijland: *Process Algebra*. Cambridge Tracts in Theoretical Computer Science, Vol. 18. Cambridge Univ. Press, Cambridge, 1991.
2. S.C.C. Blom, W.J. Fokink, J.F. Groote, I.A. van Langevelde, B. Lissner and J.C. van de Pol. μ CRL: A Toolset for Analysing Algebraic Specifications. CAV '01, pp. 250–254, LNCS 2102, 2001.
3. J.F. Groote, J. Pang and A.G. Wouters. A Balancing Act: Analysing a Distributed Lift System. Technical Report, Dept. of Software Eng., CWI, Amsterdam.
4. J.F. Groote and F. van Ham. State Space Visualization. Technical Report #0214, Dept. of Mathematics and Computer Science, Technische Univ. Eindhoven, 2002.
5. F. van Ham, H. van de Wetering and J.J. van Wijk. Visualization of State Transition Graphs, Proc. IEEE Conf. on Information Visualization '01, pp 59–66, 2001.
6. F. van Ham. Interactive Visualization of State Transition Systems. Project website at <http://www.win.tue.nl/~fvham/fsm/>.
7. S. Mauw and G.J. Veltink (eds). *Algebraic Specifications of Communication Protocols*. Cambridge Tracts in Theoretical Computer Science, Vol. 36. Cambridge Univ. Press, Cambridge, 1993.
8. G.G. Robertson, J.D. Mackinlay and S.K. Card. Cone Trees: Animated 3D Visualizations of Hierarchical Information. CHI '91 Conf. Proc., pp 189–194, 1991.