3D Layout of Reachability Graphs of Communicating Processes *

Thierry JÉRON, Claude JARD **

IRISA, Campus de Beaulieu, F-35042 RENNES Cedex, FRANCE
jeron@irisa.fr, jard@irisa.fr

Abstract. This paper presents a study about the 3D layout of some particular graphs. These graphs are finite reachability graphs of communicating processes. Some interesting semantical information which is present in the graph, such as concurrency, non-determinism, and membership of actions to processes are explicitly used in the layout. We start with the study of deterministic processes and give a conical representation which satisfies our requirements. Then we extrapolate our layout for non-deterministic processes.

1 Introduction

As graphs are used in most areas of computer science, one is often interested in viewing these graphs on a screen or a sheet of paper. This has led to a great amount of work (see for example [DBET] for a bibliography on the subject and [DBEdF*93]). Algorithms for drawing graphs often use general heuristics (for example minimization of the number of edge crossings). There is also some drawings which use the semantical information contained in particular graphs. This is the case for hierarchical drawing of some acyclic graphs [STT81].

In this paper we study the drawing of reachability graphs. These are rooted labeled graphs representing the whole behavior of distributed programs. A vertex represents a reachable global state of the program and a path from the root is labeled by a possible sequence of actions. A reachability graph carries some semantical information about the concurrency and non-determinism of pairs of actions. Viewing this information on a graphical representation is important to understand the behavior of the program. Some practical requirements have guided our choices for the representation. We want to draw the graph *on-line* i.e. while it is constructed and without affecting the already drawn part. When a new global state is constructed, its location in the layout depends only on locations of already constructed states. We also want to generate coordinates in 3 dimensions, as existing 3D tools allow to manipulate the graphs using rotations, projections and zooming. Finally, we want to *separate* states i.e. for two different states, we want to generate different 3D coordinates. Of course we are limited by the number of points on a screen. But if states have different coordinates in $\mathbb{R}^3$, using a zooming facility and rotations, we can distinguish them in 2D.

** Thierry Jéron is an Inria researcher and Claude Jard a Cnrs researcher

* This work has been supported in part by the French Ministère de l'Enseignement Supérieur et de la Recherche under the grant Trace and by the French-Israeli cooperation.

To our knowledge, there was no algorithm satisfying our requirements. There already exists visualization tools for the analysis of distributed program, but they focus on the exchange of messages in distributed programs or draw reachability graphs using general graph drawing algorithms.

The paper starts with the definition of a model of communicating processes and their reachability graphs. We distinguish two sub-classes of systems of communicating processes, deterministic and deterministic acyclic systems, corresponding to real applications which are shortly explained and motivated. We propose a first representation for deterministic acyclic systems which is straightforwardly extended to deterministic systems. We then explain how we have adapted the representation to general reachability graphs.

1.1 Reachability Graphs

Our representation can be applied to several known models of communicating processes, but for explanatory reasons we will focus on the model of finite state machines communicating through FIFO channels (CFSM for short). Though it is a simple model, it serves as a basis for several specification languages like Estelle [ISO89] and SDL [CCI87].

A CFSM system is a finite set $\{P_1, \ldots, P_n\}$ of processes. To each pair (P_i, P_j) is associated a FIFO channel $f_{i,j}$ and a finite set $M_{i,j}$ of messages that P_i can send to P_j. Each process P_i is modeled by an automaton $< Q_i, A_i, T_i, q_{0_i} >$ where Q_i is a finite set of n_i states, $q_{0_i} \in Q_i$ is the initial state, A_i is a finite set of actions, $T_i \subseteq Q_i \times A_i \times Q_i$ is the set of transitions. An action a_i is either an internal action (τ), a sending $-m$ of a message $m \in M_{i,j}$ into $f_{i,j}$ or a reception $+m$ of a message $m \in M_{j,i}$ from $f_{j,i}$.

The semantics that we consider is interleaving: if several actions of distinct processes are locally fireable, all possible interleavings must be considered as possible global behaviors. Moreover, we must also consider the non deterministic choice between local actions. This allows to consider the whole behavior of the system as a labeled transition system $\mathcal{S} =< Q, A, T, S_0 >$ where:

- $Q = Q_1 \times \ldots \times Q_i \times M_{1,2} \times \ldots \times M_{n,n-1}$. A global state $S \in Q$ is composed of an n-uple of local states $E(S) =< E_1(S), \ldots, E_n(S) >$ and a $(n^2 - n)$-tuple $C(S) =< C_{1,2}(S), \ldots C_{1,n}(S), \ldots C_{n,n-1}(S) >$ of FIFO channels contents. $S_0 =< q_{0_1}, \ldots, q_{0_N}, \emptyset, \ldots \emptyset >$ is the initial global state,
- $A = A_1 \cup \cdots \cup A_N$: an action of the system is an action of one of the processes,
- $T \subseteq Q \times A \times Q$ is the transition relation. Let S and S' be two global states and $a_i \in A_i$ be an action of P_i. The triple $t = (S, a_i, S')$ is in T if and only if $(E_i(S), a_i, E_i(S')) \in T_i$ and
  - $a_i = -m \in A_i$ (output of m from P_i to P_j through $f_{i,j}$) and
    $$\forall k \neq i,\ E_k(S') = E_k(S),\ C_{i,j}(S') = C_{i,j}(S).m,$$
    $$\text{and } \forall (k,l) \neq (i,j),\ C_{k,l}(S') = C_{k,l}(S)$$
  - $a = +m \in A_i$ (input of m in P_i from $f_{j,i}$) and
    $$\forall k \neq i, E_k(S') = E_k(S),\ m.C_{j,i}(S') = C_{j,i}(S)$$
    $$\text{and } \forall (k,l) \neq (j,i),\ C_{k,l}(S') = C_{k,l}(S)$$
  - if $a = \tau_i \in A_i$ (internal action in P_i) and
    $$\forall k \neq i,\ E_k(S') = E_k(S) \text{ and } C(S) = C(S')$$

If (S, a_i, S') is in T we say that a_i is *fireable* in S and *leads to* S' and write $S \xrightarrow{a} S'$. Let S and S' in Q and $a_1, \ldots, a_n$ a sequence of actions in A, we will write $S \xrightarrow{a_1 \ldots a_n}{}^{*} S'$ and say that S' is *reachable from* S if there exists some states $S_1 = S, S_2, \ldots S_n, S_{n+1} = S'$ such that $\forall k \in \{1 \ldots n\}$, $S_k \xrightarrow{a_k} S_{k+1}$. A state is *reachable* if it is reachable from S_0. The whole behavior of a transition system $\mathcal{S}$ can be described as a directed (possibly infinite) labeled tree. The root is labeled with the initial state S_0, vertices are labeled with reachable states, and edges are labeled with fireable actions. The *reachability graph* is obtained from this tree by merging vertices labeled with identical states. This graph is possibly infinite as FIFO channels are unbounded. But we will restrict our study to finite reachability graphs by bounding channel contents.

2 The Deterministic Case

We say that a process P_i is *deterministic* if for each state $q_i \in Q_i$ there is at most one transition (q_i, a_i, q_i'). There is no local choices between actions. The system is *deterministic* if all processes are deterministic. In this case, the transition system has the following property:

Diamond property: *If a_i and a_j are two different actions fireable in S and $S \xrightarrow{a_i} S1$, $S \xrightarrow{a_j} S_2$, then a_i is fireable in S_2 and a_j is fireable in S_1 and both lead to the same state S'.*

Proofs are given in the full paper [JJ94]. The diamond property can then be extended to sequences of actions:

Generalized diamond property: *If u and v are two different sequences of actions fireable in S and $S \xrightarrow{u}{}^{*} S1$, $S \xrightarrow{v}{}^{*} S_2$, then $v \setminus u$ is fireable in S_1 and $u \setminus v$ is fireable in S_2 and both lead to the same state S'.*

The sequence $u \setminus v$ is obtained from u by removing the actions of the greatest subsequence of v which is also a subsequence of u.

2.1 Representation in n Dimensions for Acyclic Processes

Here we consider systems of acyclic communicating finite state automata with no local choices. We find such systems in the analysis of finite execution traces of distributed programs. An observer process collects events time-stamped with a logical clock and reconstruct the causal ordering of events. The set of total orderings compatible with the observed causal ordering is represented by the lattice of consistent cuts which is distributive (see [DJR93, JJJR94] for more details). The observed behavior of each process can be seen as an acyclic automaton with no choice and the lattice as the reachability graph.

If processes are acyclic, the reachability graph is also acyclic and we show in this paragraph that it can be embedded into an n dimensional grid: vertices have integer coordinates and edges joining those vertices are parallel to the axis.

Lemma 2.1 *Let u and v be two sequences fireable in S_0 and leading to states S and S'. Then $S = S' \iff \forall i \in [1 \ldots n], |proj_i(u)| = |proj_i(v)|$, where $proj_i(u)$ is the projection of u on A_i and $|proj_i(u)|$ its length.*

The proof comes from the generalized diamond property and the fact that $proj_i(u) = proj_i(v) \iff |proj_i(u)| = |proj_i(u)|$.

Let $(\overrightarrow{u_1}, \ldots, \overrightarrow{u_n})$ be an orthonormal basis of $\mathbb{R}^n$. For each $i \in \{1, \ldots, n\}$, we assign the vector $\overrightarrow{u_i}$ to process P_i. Let S be a state reachable from S_0 by the sequence u and $f(S) = \Sigma_{i=1}^n |proj_i(u)| \overrightarrow{u_i}$. By lemma 2.1, f is an injective mapping from the set of reachable states to $\mathbb{N}^n$. If S' is reachable from S by the firing of an action a_i of process P_i, then $f(S') = f(S) + \overrightarrow{u_i}$. Thus the edge corresponding to the firing of action a_i is parallel to the $\overrightarrow{u_i}$.

2.2 3D Layout for Acyclic Processes

If $n > 3$ we must find a mapping from $\mathbb{N}^n$ to $\mathbb{R}^3$ which transforms the representation in n dimensions to a 3D representation. In addition to the on-line facility, the separability of states and the fact that we draw in 3 dimensions, we would like that the representation satisfied some constraints:

1. layer by layer construction: a state can be reachable by several sequences of actions (by commutation of independent actions) with same length. This gives a partition of states according to the length of the sequences from the root, easily obtained by a breadth first traversal. We will put all states at distance k from the root in the plane of equation $z = k$ in $\mathbb{R}^3$.
2. assignment of one direction per process: this allows to recognize that an action belongs to a given process.
3. same length to all edges: with the preceding condition, diamonds will be represented by lozenges.

Following these constraints, we have to find a family $(\overrightarrow{u_1}, \ldots, \overrightarrow{u_n})$ of vectors with same norm, one for each process, with $\overrightarrow{u_i} = (x_i, y_i, 1)$ which insures the separability of states.

The initial state S_0 is put at coordinate (0,0,0) in $\mathbb{R}^3$. Let S and S' be two states of the same layer reachable from S_0 respectively by sequences w and v (with same length). For each i, w (resp. v) contains the d_i (resp. e_i) first actions of P_i. Then $\overrightarrow{S_0S} = \Sigma_{i=1}^n d_i \overrightarrow{u_i}$ and $\overrightarrow{S_0S'} = \Sigma_{i=1}^n e_i \overrightarrow{u_i}$ with $\Sigma_{i=1}^n d_i = \Sigma_{i=1}^n e_i$.

States are separated if they satisfy $\overrightarrow{S_0S} = \overrightarrow{S_0S'} \iff S = S'$.
We only have to separate states of the same layer, thus this is equivalent to:

$$\forall (d_i)_{i=1,n}, (e_i)_{i=1,n} \in \mathbb{N}^n \text{ s.t. } \Sigma_{i=1}^n d_i = \Sigma_{i=1}^n e_i$$
$$\Sigma_{i=1}^n d_i \overrightarrow{u_i} = \Sigma_{i=1}^n e_i \overrightarrow{u_i} \iff \forall i, d_i = e_i$$
$$\text{or } \forall (f_i)_{i=1,n} \in \mathbb{Z}^n \text{ s.t. } \Sigma_{i=1}^n f_i = 0, \Sigma_{i=1}^n f_i \overrightarrow{u_i} = \overrightarrow{0} \iff \forall i, f_i = 0$$

We can strengthen this property if we consider that $\forall i, f_i \in \mathbb{Q}$ the set of rational numbers, and that we distinguish vertices on their x coordinate. We choose $-1 \leq x_i \leq 1$ and $y_i = \pm\sqrt{1 - x_i^2}$. This gives:

$$\forall (c_i)_{i=1,n} \in \mathbb{Q}^n \text{ s.t. } \Sigma_{i=1}^n c_i = 0, \Sigma_{i=1}^n c_i x_i = 0 \iff \forall i, c_i = 0$$

We show that this is satisfied if we take $x_i = t^i$ for any $t \in \mathbb{R}$ *transcendental* in $\mathbb{Q}$ and satisfying $|t| < 1$ (see [Bou81]). t is transcendental if there exists no polynomial with rational coefficients of which t is a root. For example $t = \pm 1/\pi$

or $\pm 1/e$. Let $P_n(X) = \Sigma_{i=1}^{n} c_i X^i$ with $\forall i, c_i \in \mathbb{Q}$. As t cannot be a root of P_n, we have $P_n(t) = \Sigma_{i=1}^{n} c_i t^i = 0 \iff \forall i, c_i = 0$.

Though the separability is theoretically satisfied, it is not practical because, as $|t| < 1$, $t^i \to 0$ when $i \to \infty$. Our practical solution is the following (where $D(x)$ is the decimal part of x):

$$x_1 = D(e), \qquad y_1 = \sqrt{1 - x_1^2}$$
$$x_i = (-1)^{i-1}.D(x_{i-1}.e), \; y_i = (-1)^{(i-1)div 2}.\sqrt{1 - x_i^2}, \; \forall i \in \{2 \ldots n\}$$

In this way, vectors (x_i, y_i) are distributed alternatively on the 4 sectors of the unit circle. The separability property is proved in the full paper.

The family of vectors $\overrightarrow{u_i} = (x_i, y_i, 1)$ for $i = 1 \ldots n$ is independent by linear combination with rational coefficients. As for every reachable state S we have $\overrightarrow{S_0 S} = \Sigma_{i=1}^{n} d_i \overrightarrow{u_i}$ with $\forall i, d_i \in \mathbb{N}$, and $z_i = 1$ and $x_i^2 + y_i^2 = 1$ the graph stays inside the half-cone defined by $(0, 0, 0)$ and the circle of equations $z = 1$ and $x^2 + y^2 = 1$.

Our layout program generates coordinates of vertices and edges in 3D and is interfaced with several viewing tools. We can generate different projections in 2D and produce Postscript files like in the paper. But we also generate 3D coordinates which can be used by Visage, a tool from Technion Haifa, and used for the analysis of distributed programs that allows to fly in 3D in the graph.

Complexity: For each state S, the coordinates of a successor state reachable by the firing of a_i are obtained by adding u_i to the coordinates of S. Thus, once vectors $\overrightarrow{u_i}$ have been computed, the layout is linear in the size of the graph. Computing the $\overrightarrow{u_i}$ and ordering them can be done in $O(n.log(n))$ where n is the number of processes.

Colors: In order to easily identify processes to which actions belong, we also use one color by process. Colors are generated automatically in the red/green/blue palette by a continuous function from $[1, n]$ to $[0..1]^3$.

Example: We give a simple example below with 4 communicating processes (see figure 1). The left hand graph is obtained by a simple 2D layout program which places states layer by layer according to their distance to the root. In each layer, states are distributed at a constant interval on the width of the page. The separability property is satisfied but diamonds are not clearly identified. The right hand graph is obtained by our 3D layout program and projected on a 2D space. Lozenges in 3D are then projected to parallelograms.

2.3 3D Layout for Deterministic Non Acyclic Processes

When we relax the constraint of non cyclicity we describe systems that can be obtained when automatically distributing sequential programs in an asynchronous environment (see [ACP93]). The layout of their reachability graphs allows to measure the level of parallelism for different data distributions. A bottleneck in the graph characterizes a strong synchronization of processes whereas a high level of parallelism is characterized by a wide graph.

In this case, the reachability graph can have cycles. The layout is performed during a breadth first construction of the graph. Is is almost the same as in the

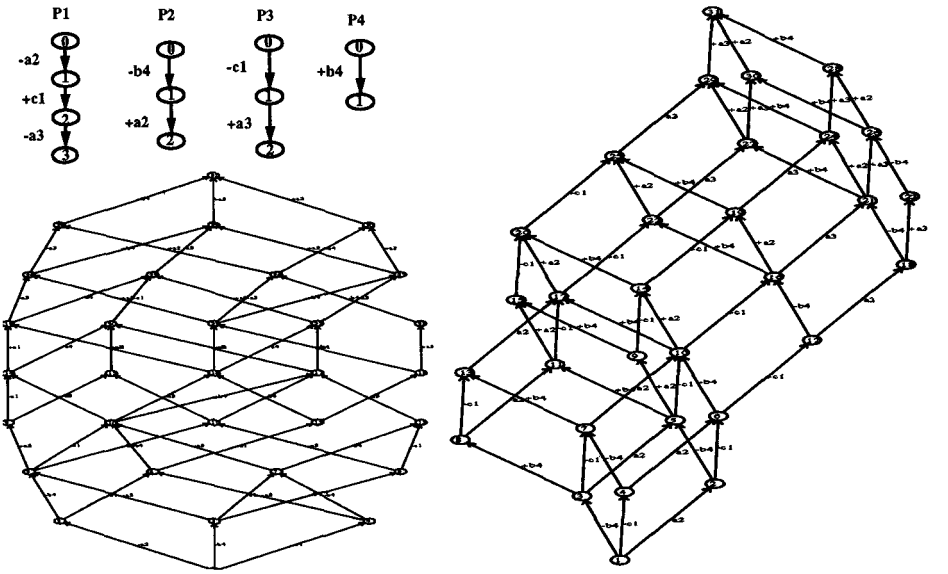

Fig. 1. Conical 3D layout of the reachability graph of a system of acyclic deterministic CFSM

acyclic case except for backward edges (which go from layer k to layer $l < k$), loop edges (which go from a state to itself), horizontal edges (which go from layer k to layer k and are not loop edges). Those edges do not follow the direction assigned to their process but are identified by the use of other colors.

3 The Non Deterministic Case

Generally communicating processes are non deterministic and can have cycles. Non determinism means that in a process several actions can be fireable in the same local state. We say that they are in conflict. As a consequence the diamond property is generally false. Moreover, two sequences composed of the same actions in a different ordering may lead to two different states. If actions are assigned fixed vectors, those states will have the same coordinates. But this contradict the separability of states. However, for pairs of concurrent actions, the diamond property is still (locally) true.

We also want that our representation of general reachability graphs preserves the representation of reachability graphs of deterministic processes.

Our choice is the following. States are still partitioned layer by layer according to their distance from the root. For each process, by static analysis, we partition the set of actions by the transitive closure of possible conflict. Let k_i be the cardinal of the biggest equivalence class of conflicting actions of process P_i and $N = \Sigma_{i=1}^{n} k_i$. We compute the $N + 1$ first vectors by the same mechanism as in the deterministic case and order them according to the value of their angle in $[0, 2\pi]$.. Let $\overrightarrow{v_1} < \overrightarrow{v_2} \ldots < \overrightarrow{v_{N+1}}$ be those ordered vectors. They are independent

in $\mathbb{R}^2$ by linear combination with rational coefficients. The $N+1^{th}$ vector will serve to separate states with same coordinates. The k_1 vectors $\{\overrightarrow{v_1}, \ldots, \overrightarrow{v_{k_1}}\}$ are assigned to P_1, the k_2 following $\{\overrightarrow{v_{k_1+1}}, \ldots, \overrightarrow{v_{k_1+k_2}}\}$ to P_2, etc ...until P_n. For each process P_i, and in each equivalence class, each action is assigned a vector among the k_i vectors assigned to P_i (with z coordinate set to 1). In this way conflicting actions have close but different directions.

This does not solve all problems as two different states may have the same location. Thus in each layer, we keep the set of coordinates of already placed states. When a new state are computed, its coordinates are compared to already computed coordinates. If they already exist (or are too close to some other coordinates), we modify them by adding $\overrightarrow{v_{N+1}}$ until there is no more conflict. In this way, all states are distinguished and the layout still corresponds to our requirements. As a consequence, when there are many conflict on coordinates, the graph tends to leave the cone in the direction of $\overrightarrow{v_{N+1}}$.

Efficiency: theoretically, the layout algorithm can be expensive because of the resolution of conflicts on coordinates. But practically there is not so much conflicts. A graph of one hundred states is drawn in less than 1 second.

Example: Adding an internal action t from 0 to 1 in the process P_1, we have introduced a non deterministic choice between t and $-a2$ in the state 0 of P_1 (see figure 2). From the initial state, these two actions are fireable from the initial state and take two different directions in the layout. The diamond property is not globally true (t and $-a2$ do not commute) but we can easily identify some diamonds in the graph.

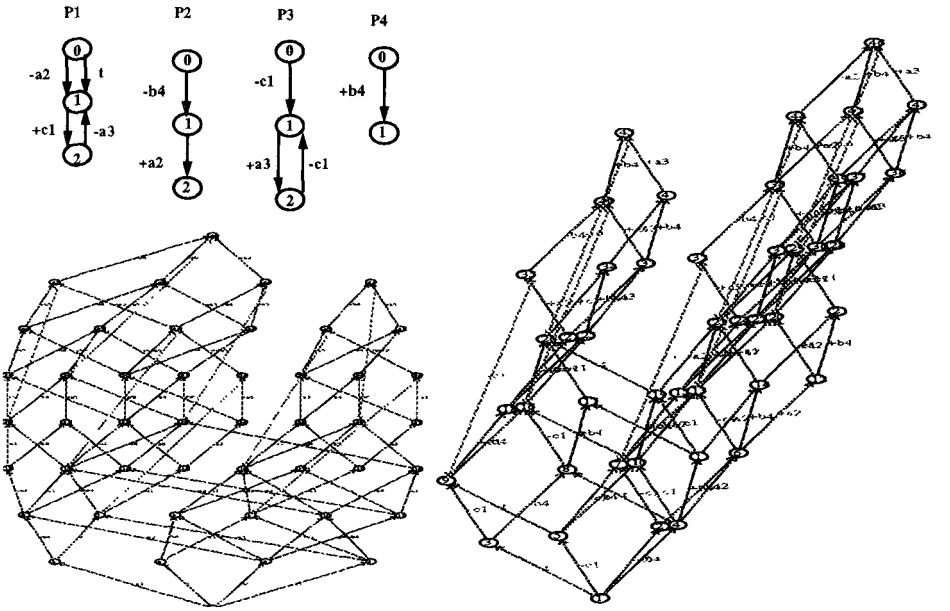

Fig. 2. 3D layout of the reachability graph of a non deterministic system

4 Conclusion

In this paper we have defined a layout program for reachability graphs of communicating processes. This program is based on the idea that semantical informations as concurrency, non determinism and membership of actions to processes are essential for the understanding of reachability graphs. It is very efficient and produces a readable layout for graphs of some tens of states. This allows us to use it as a teaching tool to understand reachability graphs. For biggest graphs (with some hundreds or thousands states) a visual analysis is more difficult. But as our layout is somehow regular, we can still identify some global structure in the graph. It is particularly true with deterministic processes. We intensively use it to compare different data distributions in the automatic distribution of sequential code and to understand lattices of consistent cuts of observed distributed executions.

Acknowledgments: we wish to thank Cyrille Bareau and Benoît Caillaud for their help in the design of our drawing algorithm.

References

[ACP93] Françoise André, Olivier Chéron, and Jean-Louis Pazat. Compiling Sequential Programs for Distributed Memory Parallel Computers with Pandore II. In J.J. Dongarra and B. Tourancheau, editors, *Environments and Tools for Parallel Scientific Computing*, pages 293–308, Elsevier Science Publishers B.V., 1993. *Also available as technical report IRISA (no. 651).*

[Bou81] N. Bourbaki. *Eléments de mathématique: Algèbre.* Masson, 1981. Chap. 4-7.

[CCI87] CCITT. *SDL, Recommendation Z.100.* 1987.

[DBEdF*93] G. Di Battista, P. Eades, H. de Fraysseix, P. Rosentiehl, and R. Tamassia, editors. *Graph Drawing '93, the ALCOM International Workshop on Graph Drawing*, Sèvres, Parc of Saint Cloud, Paris, September 1993.

[DBET] G. Di Battista, P. Eades, and R. Tamassia. Algorithms for drawing graphs: an annotated bibliography. Draft document.

[DJR93] C. Diehl, C. Jard, and J.X. Rampon. Reachability analysis on distributed executions. In M.C. Gaudel and J.P. Jouannaud, editors, *TAPSOFT*, pages 629–643, Springer–Verlag, LNCS 668, Orsay, April 1993.

[ISO89] ISO 9074. *Estelle: a Formal Description Technique based on an Extented State Transition Model.* ISO TC97/SC21/WG6.1, 1989.

[JJ94] T. Jéron and C. Jard. *3D layout of reachability graphs of communicating processes.* Technical Report 2334, INRIA, September 1994.

[JJJR94] C. Jard, T. Jéron, G.-V. Jourdan, and J.-X. Rampon. A general approach to trace-checking in distributed computing systems. In *The 14^{th} International Conference on Distributed Computing Systems, Poznan, Poland*, pages 396–403, IEEE Computer Society Press, June 1994.

[STT81] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-11(2):109–125, 1981.