Definition of global properties of distributed computer systems by the analysis of system components method

By J. R. JUST

1. Introduction

A distributed computer system (abbr. DCS) consists of a number of distinct and logically connected communicating asynchronous sequential processes. A task realization in a DCS is the result of these process activities. During the task realization a user of a system creates a virtual network of processes. The virtual network of processes consists of a set of logically connected coprocesses. Each of the coprocesses for a given virtual process is executed in another processor of a DCS.

To gain a theoretical understanding of such systems, it is necessary to find a mathematical model which reflects essential features of these systems while abstracting irrelevant details away. Such the model allows problems to be stated precisely and make them amenable to mathematical analysis.

In the papers JUST [3, 4] it has been introduced a mathematical model of a distributed computer system and a mathematical model of their input/output behaviour. We use the concept process as a basic unit in our description of a DCS, and by a mathematical model of the process we shall mean a finite-control (FC-) algorithm of MAZURKIEWICZ, PAWLAK [5]. Formally, our model is based on a notion of so called vector of coroutines. This notion has been introduced by JANICKI [1, 2], in order to describe the semantics of programs with coroutines.

The main purpose of this paper is to define the global properties of distributed computer systems by the analysis of system components (coprocesses). We would like to answer the following questions. What can we say about all possible behaviours of the whole system, if we only know the local behaviour of all particular components of a DCS? Is it possible to analyse each component independently, and then to assemble all local properties in order to get the global semantics of the virtual process executed in a DCS?

To solve these problems, we extend the theory in JUST [4], and adapt some elements of the theory from JANICKI [1].

2. The model of a distributed computer system

The mathematical model of a distributed computer system has been introduced in the paper JUST [3]. In this chapter basic facts, important to the problem examined in this paper, will be presented. For more details the reader is advised to consult JUST [3, 4].

For every $n=1, 2, ...,$ let $[n]=\{1, 2, ..., n\}$. For each alphabet Σ let $\Sigma^*=\bigcup_{i=0}^{\infty}\Sigma^i$, $\Sigma^+=\Sigma^*\Sigma$, $\Sigma^\cdot=\Sigma\cup\{\varepsilon\}$ where ε is the empty word. The remaining notation of the paper is standard.

By a model of a DCS we shall mean a 3-tuple

$$\mathrm{DCS} = (S, \mathrm{MP}, \mathrm{AL})$$

where

S is the structure of the system,
MP is the set of processes in the system,
AL is a mapping $\mathrm{AL}: \mathrm{MP}\to S$.

2.1. Structure of DCS. By the structure of DCS we mean a directed graph

$$S = (N, n_0, \mathrm{LT})$$

where

N is the set of nodes (interpreted as stations of computer network),
$n_0\in N$ is the initial node,
$\mathrm{LT}\subseteq N\times N$ is the set of edges (interpreted as transmission lines).

2.2. Processes in DCS. In order to describe the set of processes in DCS we shall introduce a mathematical object, called a matrix of coprocesses.

2.2.1. Matrix of coprocesses. By a matrix of coprocesses we mean a system

$$\mathrm{MP} = (\mathscr{A}, I_0)$$

where

$$\mathscr{A} = \{A_{ij}\}_{\substack{i\in[m]\\ j\in[n]}},\ I_0\in[m]\times[n].$$

A_{ij} is a coprocess, and I_0 indicates the start process. The set $\mathscr{A}$ can be interpreted as a matrix

$$\mathscr{A} = \begin{bmatrix} A_{11}, \dots, A_{1n} \\ \dots \\ A_{m1}, \dots, A_{mn} \end{bmatrix}.$$

Each line in the above matrix represents one process. A_{ij} is a 4-tuple which represents the j-th coprocess in the i-the process.

$$A_{ij} = (\Sigma_{ij}, V_{ij}, \sigma_{ij}, P_{ij}) \qquad \text{or} \qquad A_{ij} = (\emptyset, \emptyset, \{\varepsilon\}, \emptyset)$$

where

1) Σ_{ij} is an alphabet (of action symbols),
2) V_{ij} is an alphabet (of control symbols of A_{ij}),
3) $\sigma_{ij}\in V_{ij}$ is the start symbol of A_{ij},

4) P_{ij} is a finite subset of the set

$$\{(i,j)\}\times([m]\times[n])\times(V\times V^{\cdot})\times\Sigma_{ij}.$$

This means that P_{ij} is a finite set of 4-tuples of the form $(i\to r, j\to s, a\to b, R)$ where $i\to r\in\{i\}\times[m]$, $j\to s\in\{j\}\times[n]$, $a\to b\in V\times V^{\cdot}$, $R\in\Sigma_{ij}^{\cdot})$. P_{ij} is called the set of instructions of A_{ij}. Let $P=\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}P_{ij}$.

Each instruction consists of four parts:

1) $i\to r$ indicates the process which will be active after the execution of the instruction (the r-th process will be active);

2) $j\to s$ indicates the coprocess which will be active after the execution of the instruction;

3) $a\to b$ indicates the way of execution of the component A_{ij}. This part of the instruction indicates the current and next point of the component A_{ij}.

4) R is the action of the instruction. It is an action name. R — because of its abstract character — will mean the program, the part of the program or an activity of the opertating system.

Every matrix of coprocesses can be represented graphically by means of graphs

$$a\cdot\xrightarrow[R]{}\cdot b \quad a\cdot\overset{i,j}{\underset{R}{\Longrightarrow}}\cdot b \quad a\cdot\xrightarrow[R]{i,j}\cdot b$$

to denote instructions $(i\to i, j\to j, a\to b, R)$, $(i\to i, j\to s, a\to b, R)$ and $(i\to r, j\to s, a\to b, R)$, respectively.

Put $=\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}\Sigma_{ij}$. The set Σ is called the set of action names of the matrix MP.

Let $\mathrm{ms}=\bigtimes_{i=1}^{m}\bigtimes_{j=1}^{n}V_{ij}^{\cdot}$ ($\bigtimes$ is the cartesian product). The set $\mathrm{MS}=[m]\times[n]\times\mathrm{ms}$ is called the set of control states of MP.

Let $\mathrm{co}:[m]\times[n]\times\mathrm{ms}\to\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}V_{ij}^{\cdot}$ be the function such that, for $\alpha\in\mathrm{ms}$ and $a_{ij}\in V_{ij}^{\cdot}$, $\mathrm{co}(i,j,\alpha)=a_{ij}$.

Each $(i\to r, j\to s, a\to b, R)$ can be regarded as a relation on the set Rel(MS) defined in the following way

$$y_1(i\to r, j\to s, a\to b, R)\,y_2\Leftrightarrow(\exists\alpha,\beta\in\mathrm{ms})\ y_1=(i,j,\alpha),\ y_2=(r,s,\beta)$$

and $\mathrm{co}(i,j,\alpha)=a$, $\mathrm{co}(r,s,\beta)=b$.

The set $\mathrm{MT}=\{(i,j,\alpha)\in\mathrm{MS}|\ \mathrm{co}(i,j,\alpha)=\varepsilon\}$ is called the set of terminal control states of MP. The set $\mathrm{ST}=\mathrm{MS}\times\Sigma^{*}$ is the set of states of MP.

Let $T\subseteq\mathrm{ST}\times\mathrm{ST}$ be the relation defined by the equivalence

$$(y_1,u_1)T(y_2,u_2)\Leftrightarrow\big[\big(\exists(i\to r, j\to s, a\to b, R)\in P\big)(y_1,y_2)\in\mathrm{MS}\ \&\ u_2=u_1R\big].$$

We put $y_0=(i_0,j_0,\alpha_0)$, where

$$\mathrm{co}(i,j,\alpha_0)=\begin{cases}\sigma_{ij} & \text{for}\quad A_{ij}\neq\theta,\\ \varepsilon & \text{for}\quad A_{ij}=\theta,\end{cases}$$

(θ denotes the empty coprocess of form $(\emptyset,\emptyset,\{\varepsilon\},\emptyset)$).

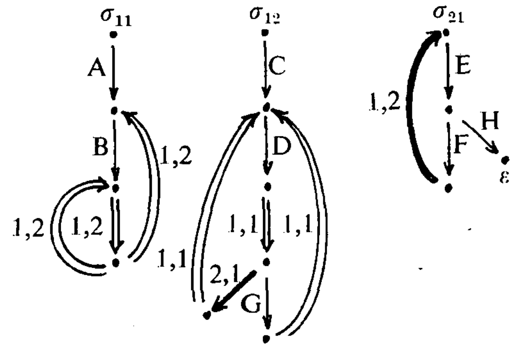
Put

$$L(\mathrm{MP}) = \{w \in \Sigma^* | (\exists y \in \mathrm{MT})(y_0, \varepsilon) T^*(y, w)\}.$$

$L(\mathrm{MP})$ is called the language generated by the matrix of coprocesses MP. The language $L(\mathrm{MP})$ represents the structure of a virtual network of processes, whereas each Σ_{ij} represents a set of names of actions (procedures) that should be executed in the framework of the (i, j)-th component of the system. This language is interpreted as a description of the semantics of the matrix MP.

Proving properties of the system of processes (in our model) is the same as proving properties of the language $L(\mathrm{MP})$. Properties of this language can be analysed by means of fixed-point methods (see JUST [3]). These methods of analysis of the matrix of coprocesses need the knowlage about the form of all components prior to the analysis. All components must be analysed together. There is a question if it is possible to analyse each component independently, and then to get the global semantics of the matrix of coprocesses. We are going to discuss this main problem of our paper in section 3.

EXAMPLE 1. Consider the system which consists of two processes, and the first process consists of two coprocesses. Let this system be represented by the following flowdiagram.



It can be proved that $L(\mathrm{MP}) = ABCD((EF \cup GB)D)^*EH$. (A method is given in JUST [4]).

2.3. Mapping AL. The mapping AL specifies an allocation of a process.

3. From local to global properties of DCS

Proving properties of a system of processes (in our model) is the same as proving properties of the language $L(\mathrm{MP})$. But the language $L(\mathrm{MP})$ does not contain much information about the structure of the matrix of coprocesses. If we know this language only, we do not know anything about the number and the form of components. Now we define a language which gives $L(\mathrm{MP})$, the number of components, and sublanguages defined by components. Note that every component can be interpreted as certain right-linnear grammar.

Let $\mathrm{MP}=(\mathscr{A}, I_0)$, where

$$\mathscr{A}=\{A_{ij}\}_{\substack{i\in[m]\\ j\in[n]}},\ I_0\in[m]\times[n]$$

and

$$A_{ij}=(\Sigma_{ij}, V_{ij}, \sigma_{ij}, P_{ij})\quad (i=1,\dots,m, j=1,\dots,n)$$

is the matrix of coprocesses.

We define the following alphabets

$$\Lambda_k=\{\lambda_{kl_1},\dots,\lambda_{kl_{mn}}\}-\{\lambda_{kk}\},\ \Lambda'_k=\Lambda_k\cup\{\lambda_k\}$$

for

$$i, r\in[m],\quad j, s\in[n]\quad \text{and}\quad k=(i,j),\quad l=(r,s).$$

Let

$$\Lambda=\bigcup_{k\in[m]\times[n]}\Lambda_k\cup\{\lambda_{k_0}\},\quad \Lambda'=\bigcup_{k\in[m]\times[n]}\Lambda'_k\quad (k_0=(i_0,j_0)).$$

The set Λ in our model represents the set of names of actions of transmissions.

Let $\lambda(\mathrm{MP})$ be the matrix of coprocesses defined as follows

$$\lambda(\mathrm{PM})=(\mathscr{A}^\lambda, I_0),\quad \text{where}\quad \mathscr{A}^\lambda=\{A_{ij}\}_{\substack{i\in[m]\\ j\in[n]}},\ I_0\in[m]\times[n].$$

$$A_{ij}^\lambda=(\Sigma_{ij}\cup\Lambda_{ij}, V_{ij}\cup\{\sigma'_{ij}\}, \sigma'_{ij}, P_{ij}^\lambda)$$

and

$$P_{ij}^\lambda=\{(i\to r, j\to s, a\to b, R\lambda_{kl})\,|\,(i\to r, j\to s, a\to b, R)\in P_{ij}\,\&\, k\neq l\}\cup$$

$$\cup\{(i\to r, j\to s, a\to b, R)\,|\,(i\to r, j\to s, a\to b, R)\in P_{ij}\,\&\, k=l\}\cup$$

$$\cup\{(i\to i, j\to j, \sigma'_{ij}\to\sigma_{ij}, \mu)\ \text{ if }\ (i,j)=I_0\ \text{ then }\ \mu=\lambda_{k_0}\ \text{ elsewhere }\ \mu=\varepsilon\}.$$

The language $L(\lambda(\mathrm{MP}))$ contains all the necessary informations about the structure of the matrix MP.

Let $h_\Lambda:(\Sigma\cup\Lambda')^*\to\Sigma^*$ be the following homomorphism

$$(\forall R\in\Sigma\cup\Lambda')h_\Lambda(R)=\begin{cases}R & \text{if } R\in\Sigma,\\ \varepsilon & \text{if } R\notin\Sigma.\end{cases}$$

Corollary. $L(\mathrm{MP})=h_\Lambda(L(\lambda(\mathrm{MP})))$.

For arbitrary $i=1,\dots,m, j=1,\dots,n$ and $k=(i,j)$ let

$$\lambda(A_{ij})=(\Sigma_{ij}\cup\Lambda'_k, V'_{ij}, \sigma'_{ij}, Q_{ij}),$$

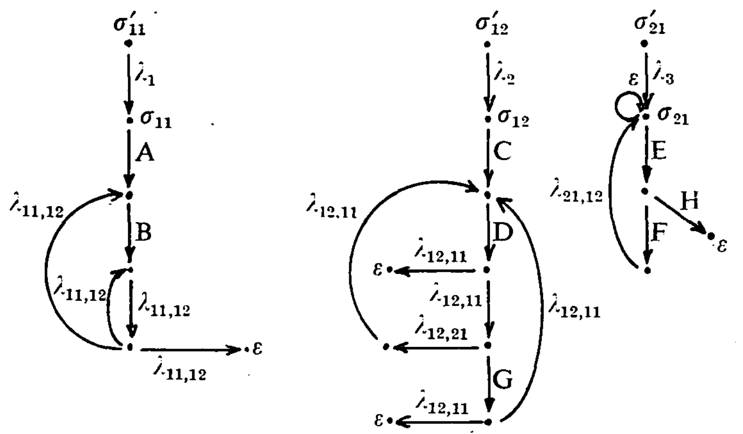
where

$$V'_{ij}=V_{ij}\cup\{\sigma'_{ij}\},\quad \sigma'_{ij}\notin V_{ij},$$

$$Q_{ij}=\{a\to Rb\,|\,\exists(i\to r,\ j\to s,\ a\to b,\ R)\in P_{ij}^\lambda\}\cup\{\sigma'_{ij}\to\lambda_k\sigma_{ij}\}.$$

Note that $\lambda(A_{ij})$ is a right-linnear grammar.

Example 2. These grammars for our matrix of coprocesses (see Example 1) can be described by the following graphs.



Let $L(\lambda(A_{ij}))$ denote the language generated by these grammars. Note that the language $L(\lambda(A_{ij}))$ not only contains an information on "actions" (elements of Σ_{ij}) of the component A_{ij}, but also on points of resumptions of another components, and an information on "actions of transmissions" as well.

For arbitrary $i, r\in[m]$, $j, s\in[n]$ and $(i\to r, j\to s, a\to b, R)$ let num: $[m]\times[n]\to\to[m\cdot n]$ be defined in the following way: num $(i,j)=(i-1)\cdot m+j$ and for $\lambda_{kl}\in\Lambda$, $k=$num (i,j), $l=$num (r,s). Let $q=m\cdot n$.

EXAMPLE 3. Languages $L(\lambda(A_{ij}))$ generated by grammars given in Example 2 are the following

$$L(\lambda(A_{11}))=\lambda_1AB\lambda_{12}((\lambda_{12}\cup\lambda_{12}B)\lambda_{12})^*\lambda_{12},$$
$$L(\lambda(A_{12}))=\lambda_2CD\lambda_{21}((G\lambda_{21}\cup\lambda_{23}\lambda_{21})D\lambda_{21})^*\lambda_{23},$$
$$L(\lambda(A_{21}))=\lambda_3(EF\lambda_{32}\cup\varepsilon)^*EH.$$

Let $\Delta_1, \ldots, \Delta_q, \Gamma_1, \ldots, \Gamma_q$ be sets defined in the following way

$$(\forall k\in[q])\quad \Delta_k=\{\lambda_{1k}, \ldots, \lambda_{k-1k}, \lambda_{k+1k}, \ldots, \lambda_q\},$$

$$\Gamma_k=\bigcup_{t\neq k, t=1}^{q}(\Sigma_t\cup\Lambda_t)-\Delta_k.$$

For arbitrary $k, l\in[q]$ and $k\neq l$ let Γ_{kl} be the following set

$$\Gamma_{kl}=\Sigma_l^*(\Lambda_l-\{\lambda_{lk}\})\Gamma_k(\Delta_k\cup\{\varepsilon\})\cup\Sigma_l^*\Delta_{lk}\cup\Sigma_l^*.$$

Let $\psi\colon(\Sigma\cup\Lambda')\to2^{(\Sigma\cup\Lambda')^*}$ be the substitution of languages defined in the following way

$$(\forall R\in\Sigma\cup\Lambda')\psi(R)=\begin{cases}R & \text{if}\quad R\in\Sigma\cup\{\lambda_{k_0}\},\\ \lambda_{k_0}\Gamma_k^*\Delta_k & \text{if}\quad R=\lambda_k\ \&\ k\in[q]-\{k_0\},\\ R\lambda_{kl} & \text{if}\quad R=\lambda_{kl}\ \&\ k\neq l.\end{cases}$$

The function ψ is called the basic semantic function. This function has been defined by Janicki [1] in order to describe the local semantics of a vector of coroutines.

A component A_{ij} of MP is called final if there exists an instruction $(i \to r, j \to s, a \to b, R)$ such that $b = \varepsilon$. The set of all final components of MP will be denoted by $\mathrm{FIN}_{\mathrm{MP}}$. We restrict our attension to the matrix of coprocesses with the property card $(\mathrm{FIN}_{\mathrm{MP}}) = 1$.

Theorem. For every matrix of coprocesses (of form defined in this paper)

$$L(\lambda(\mathrm{MP})) = \bigcap_{k=1}^{q} \psi(\lambda_k L(\lambda(A_k))),$$

where

$$A_k = A_{\mathrm{num}(i,j)} (A_{ij} \text{ is in MP}, \ i \in [m], j \in [n]).$$

The proof of the above theorem follows from considerations which have been described in [1, 2].

Example 4. Let us consider the distributed computer system which consists of three processors, connected over a communication system. These processors execute particular parts (coprocesses) of the virtual process. We know these coprocesses only. In our model they are given in the form of components of the matrix of coprocesses (see Example 1), and can be interpreted as certain right-linear grammars (see Example 2). The languages generated by these grammars, are given in Example 3.

On the basis of these languages and by taking into consideration the Theorem, we can obtain the following language

$$L(\lambda(\mathrm{MP})) = \lambda_1 AB\lambda_{12} CD\lambda_{21} ((\lambda_{12}\lambda_{23} EF\lambda_{32}\lambda_{21} \cup \lambda_{12} G\lambda_{21} B)\lambda_{12} D\lambda_{21})^* \lambda_{12}\lambda_{23} EH.$$

This language describes all possible behaviours of our distributed computer system — both computations and transmissions.

From this and from Corollary 1 it follows that $L(\mathrm{MP}) = h_A(L(\lambda(\mathrm{MP}))) = ABCD((EF \cup GB)D)^* EH$. In our model this language is interpreted as a description of the semantics of the matrix of coprocesses (the semantics of the virtual network of processes).

4. Final comment

Treating distributed systems as the superposition of sequential subsystems is the natural way of analysis and synthesis of systems. This paper is an attempt to give a formal approach to this problem. Similar problems are considered in [1, 2, 3, 4], and from a different point of view in [6].

WARSAW TECHNICAL UNIVERSITY
DEPARTMENT OF ELECTRONICS, IPE
UL. NOWOWIEJSKA 15/19 p. 230 A
00-661 WARSAW, POLAND

References

[1] Janicki, R., Analysis of coroutines by means of vector of coroutines, *Fund. Inform.*, v. 2, 3, 1979.
[2] Janicki, R., Analysis of vector of coroutines by means of components, Proc. of 2-nd Symp. on Fundamentals of Computation Theory, Berlin, 1979.
[3] Just, J. R., An algebraic model of the distributed computer system, Proc. of 5-th Conf. on the Theory of Operating Systems, Visegrad, 1979.
[4] Just, J. R., Analysis and synthesis of distributed computer systems by algebraic means., Proc. of 6-th Conf. on the Theory of Operating Systems, Visegrad, 1980.
[5] Mazurkiewicz, A., Z. Pawlak, Mathematical Foundation of Computer Science, PWN, Warsaw, 1978.
[6] Winkowski J., An algebraic approach to distributed computations, ICS PAS Reports, 377, Warsaw, 1979.

(Received Dec. 3, 1981)