General specification of multi-robot control system structures

C. ZIELIŃSKI and T. WINIARSKI

Institute of Control and Computation Engineering, Warsaw University of Technology, 15/17 Nowowiejska St., 00–665 Warsaw, Poland

Abstract. The paper deals with structuring robot control systems. The control system is decomposed into distinct agents. An agent, in general, is responsible for control of its effector, perception of the environment for the purpose of its effector control, and inter-agent communication. The behaviour of the agent is governed by its set of transition functions. The control system consists of two tiers – the upper tier is defined by the flow of information between the agents and the lower tier is defined by formal specification of each agent’s behaviour (influence on the environment, gathering sensor readings, production and consumption of the information for/from the other agents). The paper presents one of the examples of utilization of this approach. The example concerns the multi-robot drawing copying system.

Key words: robot control systems, specification of controllers, multi-robot systems.

1. Introduction

Robot control systems are usually very complex, and multi-robot system controllers are even more so. The discussion of the structures of those controllers and the way they operate requires an adequate formal language based on mathematics. There have been some attempts to formalize the subject in general (e.g., [1–3]), however, the majority of the work has concentrated on software engineering approaches to robotics [4–6], especially with the focus on robot programming frameworks [7] (e.g.: RCCL [8], KALI [9, 10], PASRO [11, 12], RORC [13, 14], MRRoC [13, 15], MRRoC++ [16, 17], G*n'oM [18, 19], DCA [20], TCA [21], TDL [22], Generis [23], OROCS [24, 25], CoolBOT [3, 26], ORCA [27, 28], Player [29–31]).

The contemporary robot system controllers are predominantly computer based. Computers are programmed using programming languages, hence they are treated as automatons accepting programs coded in those programming languages. The description of controller structure necessitates the expression of operations that the controller performs, i.e., requires the definition of semantics of those operations. For the discussion to be precise the semantics of those operations must be stated formally. It should be noted that the majority of control systems has their structure defined only on the basis of the designer’s experience. However, rational choice requires the evaluation of many conflicting criteria. Some of the usual questions that the designer must answer are:

- What information must be provided for each of the subsystems and thus what information must be exchanged between those subsystems?
- How to ensure future extensibility of the system?

To answer this type of questions the designer must have at his or her disposal a tool for formulating and evaluating decisions. Such a tool is a language for formal expression of: structure of the system, its decomposition into subsystems, definition of functions of each subsystem, communication between subsystems, evaluation of latencies introduced by each component etc. This paper provides a proposal of such a formal language. In its first part the language is described, while in its second part this language is utilized to specify the structure and operation of a drawing reproducing system controller. Unfortunately, for the lack of space, the full discussion of the considered design possibilities cannot be presented, so only the final outcome of this discussion is revealed.

The problem of describing how software based systems function is not new. Description of programming language semantics has its long history [32]. The many specific methods of defining semantics in principle can be categorized into three groups: operational semantics, denotational semantics, axiomatic (logic) semantics.

Operational semantics requires the definition of an abstract machine that accepts the instructions of the considered language. Usually this machine is capable of executing very elementary operations, that are well defined mathematically. Complex operations are defined in terms of elementary ones. Denotational semantics assigns mathematical objects (denotations) to expressions of the defined language. Those denotations describe what those expressions mean. The expressions of the original language are translated into the language of
denotations. Axiomatic semantics defines logical expressions that define the meaning of an expression of the language. One useful form of this approach defines the initial conditions for the execution of an instruction. If those conditions are fulfilled, as the result of the execution of this instruction terminal conditions are ascertained. The logical formula connects the initial and terminal conditions.

The denotational approach is translation based – its foundation is mathematical transformation. In producing a tool for the specification of robot control systems one should take into account the achievements of computer science, nevertheless, the fundamental difference between computers and robots should be kept in mind. The model of a computer is well defined and basically deterministic, while robots interact with the environment, which is modeled only approximately and cannot be treated as fully deterministic. Moreover, computers, as their name suggests, are principally used for computations, while robots are used for transforming the environment or reacting to events occurring in it. All this makes the denotational approach less attractive to our purpose. However, both the operational and axiomatic approach can be utilized to a certain extent. The choice of one of the two depends on the goal of the specification. If the aim is the definition of the control system structure and operation, operational approach is more attractive. If the goal is the definition of services provided by the system (e.g., SOA architecture), it does not reject the usefulness of the axiomatic approach favored by those who are interested in the achievements of computer science, nevertheless, the fundamental difference between computers and robots should be taken into account that the provided services must be implemented, thus the axiomatic approach still has to be redefined in terms of operations of the controller, i.e., in terms of operational semantics. Hence the approach presented in this paper is inspired by operational semantics, which is more fundamental, although more detailed than the axiomatic approach. Although this paper advocates the operational approach, because it is more relevant to the purpose of structuring robot control systems (what is at the focus of this discussion), it does not reject the usefulness of the axiomatic approach favored by those who are interested in the services provided by the system (e.g., SOA architectures [33]).

The discussion of the structures of robot control systems will be based on the concept of agent. The agents having physical bodies (e.g., robots) will be termed embodied agents. Both the operation of a single agent and the interactions between agents is of interest to us. The discussion is based on the general formalism presented in [34, 35].

Initially the necessary concepts are introduced, and subsequently an example of application of those concepts to the specification of a two-robot control system is presented.

2. An embodied agent
A multi-robot system composed of \( n_a \) agents \( a_j, j = 0, \ldots, n_a - 1 \), is considered. The internal structure of each agent \( a_j \) is presented in Fig. 1. Four distinct entities are distinguished:

- \( e_j \) – effector, i.e., a device responsible for influencing the environment (its state is obtained by reading proprioceptors), including its control hardware,
- \( R_j \) – receptors, i.e., devices gathering the information about the state of the environment (external to the agent) – subsequently processed to produce virtual sensor readings \( V_j \) (usually this information is gathered by exteroceptors, however in some cases proprioceptors can be used to detect indirectly the changes occurring in the environment, so both kinds of receptors can be a source of data for aggregation by the virtual sensors),
- \( T_j \) – transmission links, which are responsible for direct interchange of data between the considered agent \( a_j \) and the other agents,
- \( c_j \) – control subsystem – enforces a certain behaviour of the agent \( a_j \).

In this paper the symbols representing system components and their state are not differentiated, because they pertain to the same entity and context makes this differentiation obvious, whilst significantly reducing the number of symbols used.

\[
\begin{align*}
\text{effector } e_j &\quad \text{inter-agent transmission} \\
\text{transmission buffers} &\quad \text{virtual sensors } V_j \\
\text{internal variables } e^{i+1}_j &\quad \text{sensor images } v^{i+1}_j \\
\text{effector images } e^{i+1}_j &\quad \text{receptor reading/command} \\
\text{effector control/state} &\quad \text{proprioreceptors} \\
\end{align*}
\]

Fig. 1. General structure of an embodied agent

The data obtained from the exteroceptors usually cannot be used directly in motion control, e.g., control of a manipulator requires the goal location and not the bit-map delivered by a camera. In other cases a simple sensor will not suffice to control the motion (e.g., a single proximity sensor), but several such sensors deliver meaningful data about the surrounding obstacles. The process of extracting meaningful information for the purpose of motion control is named data aggregation and is performed by virtual sensors. Thus the \( k \)th virtual sensor reading obtained by the agent \( a_j \) is formed as:

\[
v_{jk} = f_{v_{jk}}(c_j, e^{i+1}_j, R_{jk}).
\]  

(1)

As the exteroceptors may have to be prompted or configured, \( c_j \) is one of the arguments of the aggregating function (1).
Moreover, a virtual sensor sometimes has its internal memory $v_{c_{jk}}$ – this is equivalent to sensoric memory in animals. Its contents is formed by an auxiliary function:

$$v_{c_{jk}} = f_{vc_{jk}}(c_j, v_{c_{jk}}, R_{jk}).$$

(2)

Obviously the $v_{c_{jk}}$ being the argument of the function $f_{vc_{jk}}$ is different from the $v_{c_{jk}}$ being the computed value of this function (the one on the left hand side of the equals sign). The former is the contents of the sensoric memory before the computation of the value of this function and the latter after the computations have been completed. Further on in the paper such distinction will be made obvious by adding a superscript representing a time stamp.

A bundle of receptors $R_{jk}$, used for the creation of the $k$th virtual sensor reading, consists of $n_r$ individual receptor readings:

$$R_{jk} = \langle r_{j_{k_1}}, \ldots, r_{j_{k_{n_r}}} \rangle,$$

(3)

where $r_{j_{k_i}}, i = 1, \ldots, n_r$, are the individual receptors taken into account in the process of forming the reading of the $k$th virtual sensor of the agent $a_j$.

The virtual sensor bundle contains $n_{v_j}$ individual virtual sensor readings:

$$V_j = \langle v_{j_1}, \ldots, v_{j_{n_{v_j}}} \rangle.$$  \hspace{1cm}  (4)

Each virtual sensor $v_{j_k}, k = 1, \ldots, n_{v_j}$, produces an aggregate reading from one or more receptors, as described by (1) and (3). Each agent $a_j$ forms and uses its own bundle $v_j$ of virtual sensors.

The first three of the four entities listed above as components of an agent $a_j$ (i.e., $c_j, V_j, T_j$) are represented in its control subsystem as images. Those images (data structures) contain parameters of the models of those components. The programmer perceives those components through those data structures, thus their names – images. The input images contain the information produced by the component for the control subsystem (denoted by a leading subscript $x$) and the output images contain the information produced by the controller for the component to utilise (subscript $y$). Diverse images (views, models) of the physical devices can be envisaged, thus creating different ontologies (An ontology in computer science is a formal representation a domain of concepts and the relationships between them – the images represent those concepts.). The control subsystem $c_j$ of the agent $a_j$ besides the above mentioned three entities contains its own internal data structures, thus the following components exist within it:

$x^{c_{ej}}$ – input image of the effector (a set of data conforming to the assumed input model of the effector in the control subsystem – it is produced by processing the input signals transmitted from the effector proprioceptors to the control subsystem, e.g., motor shaft positions, joint angles, end-effector location – they form diverse ontologies),

$x^{CV_j}$ – input images of the virtual sensors (current virtual sensor readings – control subsystem’s perception of the sensors and through them of the environment),

$x^{CT_j}$ – input of the inter-agent transmission (information obtained from other agents),

$y^{ce_j}$ – output image of the effector (a set of data conforming to the assumed output model of the effector in the control subsystem – e.g., PWM ratios supplied to the motor drivers; thus the input and output models of the effector need not be the same – and usually are not),

$y^{CV_j}$ – output images of the virtual sensors (current configuration and commands controlling the virtual sensors),

$y^{CT_j}$ – output of the inter-agent transmission (information transmitted to the other agents),

$c_{cc_j}$ – all of the other relevant variables taking part in data processing within the agent’s control subsystem.

3. General structure of images

The state of the internal data structures $c_{cc_j}$ is represented by a structure containing $n_{cc_j}$ variables:

$$c_{cc_j} = \langle c_{cc_j[1]}, \ldots, c_{cc_j[n_{cc_j}]} \rangle.$$  \hspace{1cm}  (5)

Analogically input effector image $x^{ce_j}$ consists of $n_{ex_j}$ variables:

$$x^{ce_j} = \langle x^{ce_j[1]}, \ldots, x^{ce_j[n_{ex_j}]} \rangle.$$  \hspace{1cm}  (6)

The input virtual sensor image $x^{cv_j}$ contains $n_{v_{xj}}$ individual sensor readings:

$$x^{cv_j} = \langle x^{cv_j[1]}, \ldots, x^{cv_j[n_{v_{xj}}]} \rangle,$$  \hspace{1cm}  (7)

where each of those readings has the following structure:

$$x^{cv_{jk}} = \langle x^{cv_{jk}[1]}, \ldots, x^{cv_{jk}[n_{v_{xj}}]} \rangle.$$  \hspace{1cm}  (8)

Each input transmission buffer $x^{ct_{jj'}}$ consists of $n_{tx_{jj'}}$ variables:

$$x^{ct_{jj'}} = \langle x^{ct_{jj'}[1]}, \ldots, x^{ct_{jj'}[n_{tx_{jj'}}]} \rangle.$$  \hspace{1cm}  (9)

The transmitters $ct_{jj'}$ of agent $a_j$ have received a more detailed description denoting both the owner of the transmission buffer (the first right subscript after $T$ – this is the original subscript used by the one subscript version) and the source/destination of the information (the trailing right subscript), e.g. $ct_{jj'}$ is composed of the transmission buffer of agent $a_j$ receiving information from agent $a_{j'}$: $x^{ct_{jj'}}$, or sending information to $a_{j'}$: $y^{ct_{jj'}}$. The agent $a_j$ contains as many such input transmission buffers as there are direct connections with other agents $a_{j'}$.

Generally each input transmission image of agent $a_j$ corresponds to the output transmission image of agent $a_{j'}$ and vice versa:

$$x^{ct_{jj'}} = y^{ct_{jj'}}, \quad y^{ct_{jj'}} = x^{ct_{jj'}}.$$  \hspace{1cm}  (10)
implies that there will be many partial functions that need
agent’s actions. Both selection and composition must be de-
defined formally. Usually selection is based on predicates and
composition is based on concatenation or superposition [34].
Hence, instead of a single transition function \( f_{c_j} \), \( n_f \)
partial transition functions are defined:
\[
y^{c_j+1} = m f^{i}_{c_j}(x^{c_j}), \quad m = 1, \ldots, n_f. \tag{19}
\]
Variability of agents is due to the diversity of those partial
transition functions and their different compositions. An in-
depth discussion of the possible decompositions is presented
in [34].
Each such function governs the operation of the agent for
some time. Usually this time is not defined explicitly. There
are some external to the agent factors that necessitate the
switch of the partial transition function. Such events are
detected by a Boolean valued function (a predicate) called the
terminal condition. Thus each partial transition function \( m f_{c_j} \)
is decomposed into two sub-functions: \( m f_{\tau_j} \) and \( m f'_{c_j} \). The
former expresses the terminal condition – its fulfilment stops
the repetition of computations of the latter function, i.e., the,
function responsible for the behaviour of the system within
each period \( i \rightarrow i + 1 \). This is the foundation of the general
motion instruction, which governs the activities of an agent
for the duration of the validity of transition function \( m f'_{c_j} \).

4. Transition functions
The operation of the control system of an agent can be ex-
pressed by specifying the relationship between the input and
output images. This relationship is defined in terms of transi-
tion functions. From the point of view of the system designer
the state of the control subsystem changes at a servo sampling
rate or a low multiple of that. If \( i \) denotes the current instant,
the next considered instant is denoted by \( i + 1 \). This will be
called a motion macrostep. The control subsystem uses:
\[
x^{c_j} = \langle x^{c_j}, x^{V_j}, x^{T_j} \rangle,
\]
to produce:
\[
y^{c_j+1} = \langle y^{c_j+1}, y^{V_j+1}, y^{T_j+1} \rangle. \tag{16}
\]
For that purpose it uses transition functions:
\[
\begin{align*}
  y^{c_j+1} & = f_{c_j}(x^{c_j}, x^{c_j}, x^{V_j}, x^{T_j}) \\
y^{V_j+1} & = f_{V_j}(x^{c_j}, x^{c_j}, x^{V_j}, x^{T_j}) \\
y^{T_j+1} & = f_{T_j}(x^{c_j}, x^{c_j}, x^{V_j}, x^{T_j})
\end{align*}
\tag{17}
\]
This can be written down more compactly as:
\[
y^{c_j+1} = f_{c_j}(x^{c_j}). \tag{18}
\]

5. Motion instruction
A motion instruction of each embodied agent \( a_j \) requires the
input of all arguments \( x^{c_j} \), testing of the terminal condition
\( m f_{\tau_j} \), and if it is not true, the computation of the next de-
sired values \( y^{c_j+1} \), which in turn have to be dispatched to the
appropriate components of the system. Its general form is as
follows:
\[
\text{loop} \\
  \text{// Check the terminal condition} \quad \text{if } m f_{\tau_j}(x^{c_j}) = \text{false then} \\
  \text{// Compute the next control subsystem state} \quad y^{c_j+1} := m f^{i}_{c_j}(x^{c_j}); \\
  \text{// Transmit the results} \quad y^{c_j+1} \rightarrow y^{c_j}; \quad y^{V_j+1} \rightarrow y^{V_j}; \quad y^{T_j+1} \rightarrow y^{T_j}; \tag{20}
  \text{// Wait for the next iteration} \quad i := i + 1; \\
  \text{// Determine the current state of the agent} \quad x^{c_j} \rightarrow y^{c_j}; \quad y^{V_j} \rightarrow y^{V_j}; \quad y^{T_j} \rightarrow y^{T_j}; \endloop
\]
where \( \rightarrow \) represents transfer of data. The motion instruction
starts with the test of the terminal condition, so it is assumed
that prior to the initiation of the current motion instruction all
the necessary data has been read-in by the control subsystem.
Hence the motion instruction terminates with this data being
read-in. At system initiation this data is also input.
6. Elementary behaviours

Code (20) defines the agent’s partial behaviour. Due to enormous multiplicity of possible transition functions (18) (i.e., \( m f_{e_j} \), \( m f_{r_j} \) pairs) there is no limit to the definition of those behaviours. Thus the programmer has to be supported with some guidance to facilitate the creation of useful systems. The main purpose of functions \( m f_{e_j} \) is to induce motion of the effectors. Each macrostep is divided into steps internally by the effector driver. The operation of this driver within each step \( \iota \rightarrow \iota + 1 \) is described by the following control law, which is formulated for each direction of motion separately, analogically to the Task Frame Formalism [40] or Operational Space concept [41]:

\[
E^e_{\iota+1} \left( A \right)_{\iota+1} = \left( \begin{bmatrix} \mathcal{B} \left( \Delta t = 0 \right) \left( E^e_{\iota+1} \left( A \right)_{\iota+1} - E^e_{\iota} \left( A \right)_{\iota+1} \right) + \right) \frac{\mathcal{B} \left( \Delta t = 0 \right) \left( I_{\iota+1} \right)}{\Delta t + 2 \mathcal{B} \left( \Delta t = 0 \right) \left( I_{\iota+1} \right)} \right) + \frac{\mathcal{B} \left( \Delta t = 0 \right) \left( I_{\iota+1} \right)}{\Delta t + 2 \mathcal{B} \left( \Delta t = 0 \right) \left( I_{\iota+1} \right)},
\]

where \( E \) – the frame affixed to the end-effector (tool), \( E^e \) – the superscript denotes the fact that a certain value is expressed with respect to a frame with an orientation of the frame \( E \) at instant \( \iota \), \( A \) – the subscript indicates that a certain quantity is expressed in XYZ Cartesian coordinates supplemented by an angle and axis representation of orientation, \( E^c_{\iota+1} \) – the computed generalized velocity of the end-effector in relation to the world coordinate frame for the next step \( (\iota + 1) \) expressed with respect to \( E^c_{\iota} \), \( E^d_{\iota+1} \) – the desired generalized velocity of the end-effector (set for the whole of the macrostep) in relation to the world coordinate frame for the next step \( (\iota + 1) \) expressed with respect to \( E^c_{\iota} \), \( E^t_{\iota+1} \) – desired general force for time instant \( \iota + 1 \) expressed with respect to \( E^t_{\iota} \), \( E^f_{\iota+1} \) – measured general force at time instant \( \iota \) expressed with respect to \( E^t_{\iota} \), \( I \) – desired value of reciprocal of damping, \( I \) – desired value of inertia, \( \Delta t \) – duration of a single step \( (\iota \rightarrow \iota + 1) \), \( l \) – right subscript part in square brackets denotes a coordinate of a vector. The vector components are referred to by \( x, y, z \) (linear coordinates) and \( a_x, a_y, a_z \) (angular coordinates).

It should be noted that sometimes simultaneously one form of those behaviours is expected to occur in one spatial direction, whereas another form has to be realized in another.

The three enumerated elementary behaviours are used as building blocks for constructing more elaborate functions \( f'_{e_j} \), which take into account the data obtained from virtual sensors and other agents, as presented by (20). The functions \( m f_{e_j} \) produce values that are the arguments of elementary behaviours executed in the process of transmitting the results (execution of the \( \rightarrow \) operator in code (20)).

7. Effector driver

The output effector image stores the data necessary for the computation of the control law governing the behaviour of the effector, i.e., the manipulator in this specific case. The agent’s control system forms commands for the Effector Driver. Each transmission of the output image to its respective component of the agent defines the behaviour of that component during the next macrostep, so it also defines the behaviour of the manipulator by delivering the parameters to the control law implemented in the effector driver. Each macrostep is divided into steps internally by the effector driver. The operation of this driver within each step \( \iota \rightarrow \iota + 1 \) is described by the following control law, which is formulated for each direction of motion separately, analogically to the Task Frame Formalism [40] or Operational Space concept [41]:

\[
E^e_{\iota+1} \left( A \right)_{\iota+1} = \left( \begin{bmatrix} \mathcal{B} \left( \Delta t = 0 \right) \left( E^e_{\iota+1} \left( A \right)_{\iota+1} - E^e_{\iota} \left( A \right)_{\iota+1} \right) + \right) \frac{\mathcal{B} \left( \Delta t = 0 \right) \left( I_{\iota+1} \right)}{\Delta t + 2 \mathcal{B} \left( \Delta t = 0 \right) \left( I_{\iota+1} \right)} \right) + \frac{\mathcal{B} \left( \Delta t = 0 \right) \left( I_{\iota+1} \right)}{\Delta t + 2 \mathcal{B} \left( \Delta t = 0 \right) \left( I_{\iota+1} \right)}.
\]

This is executed by the position axis-controller after transformation by the inverse kinematics procedure. A detailed presentation of the driver is contained in [42].

8. Example: copying drawings by a multi–robot system

The utilization of the above mentioned formal considerations will be presented here on an example of the specification of a controller for a robot system reproducing the taught–in drawings. Both the teach–in phase and the reproduction phase will be specified.
The experimental setup (Fig. 2) consists of two modified IRb-6 manipulators with additional active degree of freedom located in the wrist [43] and force/torque sensors, conveyor, PC computers connected by an Ethernet network supervised by the QNX Neutrino real-time operating system.

Reproducing a drawing by a robot has attracted the attention of other researchers [39]. In our investigations [44–48] the force sensor is used to manually guide the robot holding a pen through the motions producing a drawing and then to reproduce it either by the same robot (Fig. 7) or simultaneously by two robots. Only the latter is specified here.

The teach-in process is conducted by an operator leading the robot arm and thus producing the original drawing. The reproduction phase is done automatically by two robots.

The force sensors play a dual role. On the one hand, they are involved in continuous limb control, thus they are treated as proprioceptors, and, on the other hand, they detect events occurring in the environment, thus they behave as an exteroceptor. The latter behavior requires the creation of virtual sensors.

Each virtual sensor monitors the state of the drawing process. To do so it contains a finite state automaton (Fig. 3) that monitors the current state of the pen. The force and position measurements are the input obtained directly from the Effector Driver. The current state is memorized in the intertial virtual sensor memory $v_{ij}$. Drawing starts in the Above paper state. Arc $A$ is activated when a downward jerk is detected. The automaton state changes to Lowering. The $B$ arc is activated when the impact is detected, leading to the Paper surface state. The $C$ arc is associated with an upward jerk and the automaton changes its state to Lift-off. Then in the teach-in phase the system switches immediately to the Above paper state (arc $D$). Thus by traversing the states of this automaton the virtual sensor is able to notify the agent whether the pen tip is currently on the paper surface or above it, and so how should it behave during the drawing reproduction phase. The $v_{ij} \in \{\text{Above paper, Lowering, Paper surface, Lift-off}\}$ informs in which state is the automaton, what reflects the current state of drawing.

The experimental system [44] consists of two robots. Both manipulators reproduce drawings, while only one of the manipulators is used to teach them. We started with single robot tasks and gradually shifted out attention to more complex multi-robot tasks. The structure of a multi-robot drawing system is very similar to the structure of the Rubik’s cube puzzle solving system [36] (however the former does not require vision, whereas the latter does).
following. First the data structures on which each agent operates are presented and then the transition functions defining the behaviour of each agent.

8.1. Structure of the images. In the following the variables, described in general by (5)–(14), are presented for all of the mentioned agents.

Images of the coordinator – agent \(a_0\). The input transmission image \(x_{CT_{a0}}\) from agent \(a_1\) consists of three variables. Thus \(n_{x,v_{a0}} = 3\). The input transmission image \(x_{CT_{a2}}\) from agent \(a_2\) consists of a single variable \((n_{x,v_{a2}} = 1)\). The values of those images are defined within agents \(a_1\) and \(a_2\).

The input transmission image \(x_{CT_{a0}}\) originates with the operator interface \((n_{x,v_{a0}} = 1)\):
\[
x_{CT_{a0}}^{i} = o_{h}^{i},
\]
where \(o_{h}^{i} \in \{continue, trigger\}\) is the signal sent by an operator. Here the operator is treated as the fourth agent \(a_h\), its internal structure is not elaborated (for obvious reasons).

The input image of the virtual sensor \(x_v^{0}\) is not used \((n_{v,v_{a0}} = 0)\), thus
\[
x_v^{0} = 0.
\]
The output image of transmission image \(y_{CT_{a0}}\) and the output images of the virtual sensor \(y_v^{0}\) are not used \((n_{y,v_{a0}} = 0, n_{v,v_{a0}} = 0)\), thus
\[
y_{CT_{a0}} = 0,
y_v^{0} = 0.
\]

Thus there is no need to define \(m_{f_{CT_{a0}}}\) and \(m_{f_{v_{a0}}}\).

All of the data that must be memorized is extracted from input images \(x_c^{0}\) and is stored in \(c_{v_{a0}}\), where \(n_{c,v_{a0}} = 6\). An agent stores the trajectory which is memorized during teach−in process of a single manipulator and then reproduced during reproduction phase in two manipulator system. The description of the trajectory consists of the components of the first four variables presented below:

- \(c_{v_{a0}}[p]\) – the list of the manipulator end−effector velocities \(p = 1, \ldots , n\),
- \(c_{v_{a0}}[p]\) – the list of the states of drawing as defined by the graph in Fig. 3,
- \(c_{v_{a0}}[p]\) – the number (label) of the current node of the trajectory \((p)\),
- \(c_{v_{a0}}[p]\) – the current number of trajectory nodes \((n)\),
- \(c_{v_{a0}}[p]\) – the current end−effector pose obtained from \(x_{CT_{a0}}^{i}\),
- \(c_{v_{a0}}[p]\) – a certain time instant obtained from \(x_{CT_{a0}}^{i}\),

The current index \(p\) used to index \(c_{v_{a0}}[p]\) and \(c_{v_{a0}}[p]\) lists indicates the currently processed node of the trajectory, while \(c_{v_{a0}}[p]\) is the total number of trajectory nodes and \(c_{v_{a0}}[p]\) is the number of the currently memorized or reproduced node (Fig. 4).

Fig. 4. Data structures used for memorizing the trajectory during the teach−in process.

Output transmission buffers \(y_{CT_{a1}}\) to agents \(a_1\) and \(a_2\) consists of ten elements each, thus \(n_{y,v_{a1}} = 10, j = 1, 2\) as defined by (33).

Images of agents \(a_1\) and \(a_2\). The input virtual sensor images \(x_v^{0}\) acquire the drawing state that is produced by the virtual sensor and is defined by the graph presented in Fig. 3 \((n_{v,v_{a0}} = 1)\):
\[
x_v^{0} = v_{j}^{0}.
\]

In this case the virtual sensor \(v_{j}^{0}\) aggregates information from the proprioceptors. Technically this information is delivered by the Effector Driver.

The general formula (1) assumes the following form \((j = 1, 2)\):
\[
v_{j}^{0} = f_{v_{j}}(v_{j-1}^{0}, R_{j}^{0}) = f_{v_{j}}(v_{j-1}^{0}, R_{j}^{0}) = 0, T_{m_{j}}, E_{j}, F_{j}^{i},
\]
where \(0, T_{m_{j}}\) – the current manipulator tool \(E\) with respect to the base frame \(0\) obtained by agent \(a_{j}\), \(v_{j-1}^{0}\) – the state of the pen attached to the effector of agent \(a_{j}\) produced by the virtual sensor of this agent and (2) assumes the form:
\[
v_{j}^{0} = f_{v_{j}}(v_{j-1}^{0}, R_{j}^{0}) = f_{v_{j}}(v_{j-1}^{0}, 0, T_{m_{j}}, E_{j}, F_{j}^{i}),
\]
where \(v_{j}^{0}, v_{j-1}^{0} \in \{Above\ paper, Lowering, Paper surface, Lift-off\}\). The embodied agents \(a_1\) and \(a_2\) do not send any commands to their virtual sensors, hence the output images of virtual sensors are not used \((n_{v,v_{j}} = 0)\):
\[
y_{v}^{0} = 0.
\]

Thus there is no need to define \(m_{f_{v_{a0}}}\) \((j = 1, 2)\).

The input effector image \(x_{e_{1}}\) contains \(n_{e_{1}} = 2\) data items:
\[
x_{e_{1}}^{i} = 0, T_{m_{1}},
\]
\[
x_{e_{1}}^{i} = i.
\]

The first is the record of the current end−effector location and the second of the current time. The input effector image \(x_{e_{2}}\) is not used \((i.e., n_{e_{2}} = 0)\):
Input transmission buffers $x_{CT_{0}}$ of agents $a_1$ and $a_2$ consist of ten elements ($n_{xT_{0}} = n_{xT_{0j}} = 10$, $j = 1, 2$), the variables holding the effector command sent by the coordinator $a_0$.

The output effector images $y_{C_{ej}}$ contain $n_{x ej} = n_{xT_{0j}} = n_{xT_{0j}} = 10$, $j = 1, 2$, components of the command sent to the effector, initially prepared by the coordinator.

The agent $a_1$ sends both the input effector image and the input virtual sensor image to the coordinator. This is done through the output transmission images: $y_{CT_{10}}$, consisting of three variables ($n_{xT_{10}} = n_{x v1} = n_{xT_{0j}} = 3$).

The agent $a_2$ sends its input virtual sensor image to the coordinator. This is done through the output transmission images: $y_{CT_{20}}$, consisting of a single variable ($n_{xT_{20}} = n_{x v2} = n_{xT_{02}} = 1$).

\[ y_{C_{T_{20}}} = v_{fi}. \]  
\( (32) \)

8.2. Transition functions and terminal conditions. In the following the transition functions essential for teach–in and reproduction subtasks are presented. Those functions take as arguments the variables defined in Subsec. 8.1 and produce reproduction subtasks are presented. Those functions take as arguments the variables defined in Subsec. 8.1 and produce

- $m = 1, 2$ – drawing teach–in,
- $m = 3, 4, 5, 6, 7$ – drawing reproduction.

![Fig. 5. Single block of the automaton presented in Fig. 6](image)

**Fig. 5.** Single block of the automaton presented in Fig. 6

![Fig. 6. The correspondence between the state of the automaton presented in Fig. 3 and the transition functions and terminal conditions responsible for the execution of the drawing task.](image)

**Fig. 6.** The correspondence between the state of the automaton presented in Fig. 3 and the transition functions and terminal conditions responsible for the execution of the drawing task. (a) Drawing teach–in phase, (b) Drawing reproduction phase

The coordinator $a_0$ produces the contents of both output transmission images $y_{C_{T_{0}}}$ by using $m_{f_{CT_{10}}}$ ($j = 1, 2$), ($m = 1, . . . , 7$):

\[ m_{f_{CT_{10}}}(x_{C_{1j}}) \triangleq \begin{cases} y_{C_{T_{0j}}}^{c+1} = m_{y_{C_{T_{0}}}^{c+1}} \\ y_{C_{T_{0j}}}^{c+1} = m_{y_{C_{T_{0}}}^{c+1}} \end{cases} \]  
\( (33) \)

where $W$ – frame affixed to the manipulator wrist, $b$ – type of elementary behaviour, $n_s$ – the number of steps that the Effector Driver divides the macrostep into, $n_Q$ – the step number in which the Effector Driver communicates with the control subsystem, $\mu_1 = \text{TCIM-velocity}$ – the choice of task coordinates interpolated motion specified in terms of velocity, $g$ – the distance between the gripper jaws (in this example it is disregarded), $W E_{T_{di}}$ – the manipulator tool $E$ with respect to the wrist frame $W$.

The symbol $\triangleq$ should be read as: “is defined as”. In the case of the drawing reproduction phase the parameters sent to both embodied agents are exactly the same in each interval (macrostep).

Embodied agent $a_1$ employs the functions $m_{f_{CT_{10}}}$ ($m = 1, . . . , 7$) to transfer data to the coordinator.

\[ m_{f_{CT_{10}}}(x_{C_{1j}}) \triangleq \begin{cases} y_{C_{T_{0j}}}^{c+1} = x_{C_{1j}}^{c+1} \\ y_{C_{T_{0j}}}^{c+1} = x_{C_{1j}}^{c+1} \end{cases} \]  
\( (34) \)

Similarly embodied agent $a_2$ employs the functions $m_{f_{CT_{20}}}$ ($m = 1, . . . , 7$) to transfer data to the coordinator. This function simply copies the input virtual sensor image $x_{C_{1j}}$ to the output transmission image $y_{CT_{20}}$:

\[ m_{f_{CT_{20}}}(x_{C_{1j}}) \triangleq \begin{cases} y_{C_{T_{2j}}}^{c+1} = x_{C_{1j}}^{c+1} \end{cases} \]  
\( (35) \)

Analogically, for both embodied agents $a_1$ and $a_2$, the function $m_{f_{C_{ej}}}$ ($m = 1, . . . , 7$) produces the Effector Driver command $y_{C_{ej}}$ by copying the input transmission image $x_{C_{T_{ej}}}$ containing the command that had been previously prepared by the system coordinator $a_0$:

\[ m_{f_{C_{ej}}}(x_{C_{ej}}) \triangleq \begin{cases} y_{C_{ej}}^{c+1} = x_{C_{ej}}^{c+1} \quad \text{for} \quad j = 1, 2 \end{cases} \]  
\( (36) \)

where $j = 1, 2$.

In the following the transition functions essential for teach–in and reproduction subtasks are presented.

Bull. Pol. Ac.: Tech. 58(1) 2010
Drawing teach-in phase. As it was previously mentioned, the teach-in process for multiple robots utilizes a single manipulator. The manipulator commanded by agent $a_2$ stands still (Table 1). The manipulator used in the teach-in process ($a_1$) is compliant in linear directions, but its orientation is fixed (Table 2). It should be reminded here that the primary job of the manipulator is to behave as required. Transition functions obviously produce other values, governing inter-agent transmissions, management of internal data structures etc.

Table 1

<table>
<thead>
<tr>
<th>$c$</th>
<th>$m_b$</th>
<th>$m_{F_a}$</th>
<th>$m_{i_{(\text{A})}^i_{(\text{A})}^i}$</th>
<th>$m_I$</th>
<th>$m_{B_2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$y$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$z$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$a_x$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$a_y$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$a_z$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
</tbody>
</table>

The function $f_{c_{\text{eq}}}$ initiates the internal variables:

$$1 f_{c_{\text{eq}}}(x_{\text{eq}}^0) \triangleq \begin{cases} c_{\text{eq}[1]} = 1 \\ c_{\text{eq}[2]} = 0 \\ c_{\text{eq}[3]} = x_{\text{eq}[3]}^0 \end{cases}.$$  \tag{37}

The current node of the trajectory is set to 1. This phase finishes when the pen touches the paper:

$$1 f_{\text{eq}}(x_{\text{eq}}^0) \triangleq \begin{cases} 0 & \text{for } x_{\text{eq}[1]} = \text{Paper surface} \\ 1 & \text{otherwise} \end{cases}.$$  \tag{38}

The next phase is the teach-in process itself (recording of the drawing). Now the manipulator is moved in the same way, but the motion trajectory is recorded by memorizing the velocity of the pen tip $\dot{r}_{(\text{A})}^i$, on the surface of the paper ($x-y$ coordinates), at constant intervals of time ($n_{s_1} = 20$ ms).

<table>
<thead>
<tr>
<th>$c$</th>
<th>$m_b$</th>
<th>$m_{F_a}$</th>
<th>$m_{i_{(\text{A})}^i_{(\text{A})}^i}$</th>
<th>$m_I$</th>
<th>$m_{B_1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$c$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0.005$</td>
</tr>
<tr>
<td>$y$</td>
<td>$c$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0.005$</td>
</tr>
<tr>
<td>$z$</td>
<td>$c$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0.005$</td>
</tr>
<tr>
<td>$a_x$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$a_y$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$a_z$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
</tbody>
</table>

The function $2 f_{c_{\text{eq}}}$ is defined as:

$$2 f_{c_{\text{eq}}}(x_{\text{eq}}^0) \triangleq \begin{cases} c_{\text{eq}[1]} = 1 \\ c_{\text{eq}[2]} = 0 \\ c_{\text{eq}[3]} = x_{\text{eq}[3]}^0 \end{cases}.$$  \tag{39}

<table>
<thead>
<tr>
<th>$c$</th>
<th>$m_b$</th>
<th>$m_{F_a}$</th>
<th>$m_{i_{(\text{A})}^i_{(\text{A})}^i}$</th>
<th>$m_I$</th>
<th>$m_{B_2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$y$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$z$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$a_x$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$a_y$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$a_z$</td>
<td>$u$</td>
<td>$0$</td>
<td>$-20$</td>
<td></td>
<td>$0$</td>
</tr>
</tbody>
</table>

The function $2 f_{c_{\text{eq}}}$ is defined as:

$$2 f_{c_{\text{eq}}}(x_{\text{eq}}^0) \triangleq \begin{cases} c_{\text{eq}[1]} = 1 \\ c_{\text{eq}[2]} = 0 \\ c_{\text{eq}[3]} = x_{\text{eq}[3]}^0 \end{cases}.$$  \tag{40}

The function $2 f_{c_{\text{eq}}}$ is defined as:

$$2 f_{c_{\text{eq}}}(x_{\text{eq}}^0) \triangleq \begin{cases} c_{\text{eq}[1]} = 1 \\ c_{\text{eq}[2]} = 0 \\ c_{\text{eq}[3]} = x_{\text{eq}[3]}^0 \end{cases}.$$  \tag{41}

Drawing reproduction phase. The second phase of the task execution consists in the reproduction of the memorized drawing. Now the coordinator sends exactly the same commands to both manipulators. At the beginning the operator moves two compliant manipulators to the initial locations, in which the memorized drawing has to be reproduced (Table 2). The function $3 f_{c_{\text{eq}}}$ is an identity function retaining the previously memorized data:

$$3 f_{c_{\text{eq}}}(x_{\text{eq}}^0) \triangleq \begin{cases} c_{\text{eq}[1]} = c_{\text{eq}[1]}^0 \\ c_{\text{eq}[2]} = c_{\text{eq}[2]}^0 \\ c_{\text{eq}[3]} = c_{\text{eq}[3]}^0 \\ \cdots \\ c_{\text{eq}[6]} = c_{\text{eq}[6]}^0 \end{cases}.$$  \tag{42}

In the initial location the pen should be above the paper surface. Then the operator signals the system to start the automatic reproduction process:

$$3 f_{c_{\text{eq}}}(x_{\text{eq}}^0) \triangleq \begin{cases} c_{\text{eq}[1]} = c_{\text{eq}[1]}^0 \\ c_{\text{eq}[2]} = c_{\text{eq}[2]}^0 \\ c_{\text{eq}[3]} = c_{\text{eq}[3]}^0 \\ \cdots \\ c_{\text{eq}[6]} = c_{\text{eq}[6]}^0 \end{cases}.$$  \tag{43}

First, both pens are moved down (the Lowering state (Table 3a). The function $4 f_{c_{\text{eq}}}$ is an identity function retaining the previously memorized data:

$$4 f_{c_{\text{eq}}}(x_{\text{eq}}^0) \triangleq \begin{cases} c_{\text{eq}[1]} = c_{\text{eq}[1]}^0 \\ c_{\text{eq}[2]} = c_{\text{eq}[2]}^0 \\ c_{\text{eq}[3]} = c_{\text{eq}[3]}^0 \\ \cdots \\ c_{\text{eq}[6]} = c_{\text{eq}[6]}^0 \end{cases}.$$  \tag{44}
b) horizontal motion above the surface of the paper \((m = 7)\)

<table>
<thead>
<tr>
<th>(c)</th>
<th>(b)</th>
<th>(d)</th>
<th>(d^+(A)_d)</th>
<th>(f)</th>
<th>(\mathcal{G})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x)</td>
<td>(u)</td>
<td>(-)</td>
<td>(\delta^+(A)_d)</td>
<td>(-)</td>
<td>(-)</td>
</tr>
<tr>
<td>(y)</td>
<td>(u)</td>
<td>(-)</td>
<td>(\delta^+(A)_d)</td>
<td>(-)</td>
<td>(-)</td>
</tr>
<tr>
<td>(z)</td>
<td>(c)</td>
<td>(1)</td>
<td>(20)</td>
<td>0.005</td>
<td>(-)</td>
</tr>
<tr>
<td>(a_x)</td>
<td>(u)</td>
<td>(-)</td>
<td>(0)</td>
<td>(-)</td>
<td>(-)</td>
</tr>
<tr>
<td>(a_y)</td>
<td>(u)</td>
<td>(-)</td>
<td>(0)</td>
<td>(-)</td>
<td>(-)</td>
</tr>
<tr>
<td>(a_z)</td>
<td>(u)</td>
<td>(-)</td>
<td>(0)</td>
<td>(-)</td>
<td>(-)</td>
</tr>
</tbody>
</table>

The function \(5\, f'_{c_{0i}}\) increments the current time instant for the memorized trajectory node:

\[
5\, f'_{c_{0i}}(x_{0i}^{+}) = \begin{cases} 
  c_{0i}^{i+1} = c_{0i}^i + 1 & \text{for } l = 1, 2, 4, 5, 6 \\
  c_{0i}^{i+1} = c_{0i}^i & \text{for } l \neq 1, 2, 4, 5, 6
\end{cases}
\]

(47)

The drawing of a single segment lasts until the pen tip reaches the location in which an upward jerk was recorded in the teach-in phase or reproduction is finished:

\[
5\, f_{\tau_0}(x_{0i}^{+}) = \begin{cases} 
  0 & \text{for } c_{0i}^{i+1} \neq \text{Paper surface} \land c_{0i}^{i+1} \neq \text{Paper surface} \\
  1 & \text{otherwise}
\end{cases}
\]

(48)

Then the pen is raised above the paper (Table 3b). The function \(6\, f'_{c_{0i}}\) memorizes the initial time instant of the current transition function execution (beginning of the lift-off operation):

\[
6\, f'_{c_{0i}}(x_{0i}^{+}) = \begin{cases} 
  c_{0i}^{i+1} = x_{0i}^{i+1} & \text{for } i = i_0 \\
  c_{0i}^{i+1} = c_{0i}^i & \text{for } i \neq i_0
\end{cases}
\]

(49)

The operation is executed until the desired time elapses:

\[
6\, f_{\tau_0}(x_{0i}^{+}) = \begin{cases} 
  0 & \text{for } x_{0i}^{i+1} < i_d \\
  1 & \text{otherwise}
\end{cases}
\]

(50)

where \(i_d\) is the desired duration.
The trajectory that the pen tip traverses above the paper surface in the horizontal plane is an accurate copy of the memorized trajectory in the same plane (46). In the reproduction phase the motions executed in the Above paper and Lift-off states are fully position controlled, whilst in the other state hybrid position-force control is utilized. The Above paper motions are executed in a horizontal plane (the \( z \) coordinate is kept constant (Table 4b), even if it varied during the teach-in phase). The function \( \tau f'_{c_{\tau_0}} \) increments the current time stamp of the memorized trajectory pose:

\[
\tau f'_{c_{\tau_0}}(x_{c_{\tau_0}}) \triangleq f'_{c_{\tau_0}}(x_{c_{\tau_0}}).
\]  

(51)

The motion above the paper lasts until the pen tip reaches the location in which an impact was recorded in the teach-in phase.

\[
\tau f_{\tau_0}(x_{c_{\tau_0}}) \triangleq \begin{cases} 
0 & \text{for } c_{\tau_0[2]} \neq \text{Paper surface} \land c_{\tau_0[3]} \neq p = c_{\tau_0[4]} \\
1 & \text{otherwise} 
\end{cases}
\]  

(52)

Then the pen can move down again to start reproducing of the following segment.

### 8.3. Drawing reproduction using a conveyor

The multi-robot drawing task was also executed in another configuration. The motion of the manipulators in the \( y \) direction was substituted by the motion of a conveyor on which the two drawing papers were located. So instead of moving the robots in the \( y \) direction the conveyor shifted the paper in that direction. Hence, the system consisted of three motion inducing devices, each capable of independent motion. Thus a three effector MRROC++ based system was created. The tests demonstrated that the so extended system also works correctly.

#### 8.4. Experimental results

The presented formal specification was used as the description of the controller of a robot capable of reproducing taught-in drawings. The controller was implemented using the MRROC++ [36, 48, 49] robot programming framework. This controller was used in the below described experiments. Figure 8 presents the three dimensional trajectories of the end-effector motion during teach-in and reproduction of the six feathers of an arrow drawn by the operator (Fig. 7). A visible difference between the graphs is caused by the way the pen moves up and down and above the paper. The operator makes unconstrained moves, hence the trajectory above the paper is uneven in the vertical direction. The reproduction algorithm produces exact horizontal motions, thus the trajectories above the paper are horizontal. This is evident in the graph in Fig. 9. The plots obtained for both effectors during the reproduction phase are very similar, thus only the plots for one of the effectors are presented here.

There are four segments of the trajectory marked as: 1, 2, 3 and 4 in the graphs in Figs. 9 and 10. All the four segments occur while drawing each feather of an arrow: 1 – motion on the paper surface, 2 – pen tip lift-off, 3 – motion above the paper, 4 - lowering of the pen tip. The symbol “∗” draws attention to the fragment of the plot representing the impact caused by the pen tip hitting the surface of the paper. During the whole of the teach-in phase and in segment 1 of the reproduction phase, the Effector Driver is commanded to reach the vertical force of \( 1 \) N, however, in the segments: 2, 3 and 4 of the reproduction phase, the motion is purely position controlled. Experiments show that the applied algorithms are robust enough to execute the whole task correctly.

![Fig. 8. Three dimensional motion trajectory during copying drawings](image-url)
9. Conclusions

The paper presents a formal approach to the specification of controllers executing diverse tasks. This approach assumes that a multi-robot system is composed of embodied agents, where each such agent has its effector, receptors and a capability to exchange information with other such agents.

The structure is expressed in terms of interconnections between the agents constituting the system, while its operation is described in terms of transition functions governing the actions of each agent. This part of the presented approach is general. Every system can be described in such a way. However, the paper goes deeper into the description of a particular system containing manipulators interacting with their environment. In this case three elementary behaviours have been distinguished and a control law enabling the implementation of those behaviours has been formulated. The presented method of system specification falls into the category of top-down methods, where the general description is refined going into ever more detailed description. The general approach has been exemplified by specifying the operation of a two effector robot system capable of reproducing drawings. This specification was subsequently used as the basis for the implementation of the system. MRRRC++ robot programming framework was used as an implementation tool for it. This design procedure has been used also for the implementation of other systems, e.g.: two-handed system solving a Rubik’s cube puzzle, two-effector haptic device, where one of the arms was used as a master and the other as a slave device. In all of those cases the proposed design procedure led to a quick and effective result.

Acknowledgements. The authors gratefully acknowledge the support of the Ministry of Science and Higher Education grant NN514 1287 33.

REFERENCES

General specification of multi-robot control system structures


