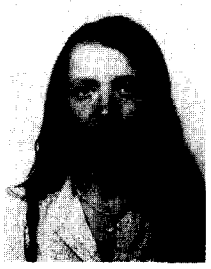
Forum

ADA: A Language for Robot Programming?

Giuseppina Gini and Maria Gini
Dept. of Electronics, Politecnico, Milano, Italy

Robot programming languages are emerging from their experimental stage and entering an assessment phase. Their main features are illustrated and a parallel with ADA is proposed. The comparison is positive for ADA, in the sense that ADA provides most of the required capabilities. The ability of reasoning on object models and taking decisions will play an increasing role in the future. In this case the role of ADA would possibly change and its interest as robot programming language decrease.

Keywords: Programmable automation, robot teaching and programming, programming systems for robots, high level languages, ADA



Giuseppina C. Gini received her doctoral degree in physics from the State University in Milano, Italy, in 1972. Since then she has been research fellow at the Electronics Department of Politecnico, Milano, Italy, where she is now Senior Research Associate. She spent more than two years at the Artificial Intelligence Laboratory of the Stanford University where she worked in the Hand-Eye project. She has published papers in the area of programming languages for artificial intelligence, assembly automation, robot programming and program construction.

Maria L. Gini received her doctoral degree in physics from the State University in Milano, Italy, in 1972. She worked as a Research Fellow from 1972 at the Electronics Department of the Politecnico, Milano, Italy. During this period she spent more than two years at the Artificial Intelligence Laboratory of the Stanford University. At present she is Senior Research associate at the Politecnico in Milano. Her research interests are in the field of artificial intelligence and its applications to robotics. She is author of several publications on those subjects.

North-Holland Publishing Company
Computers in Industry 3 (1982) 253–259

1. Introduction

During recent years a big effort in time and resources has been spent in developing advanced programming languages and systems for robots. Several approaches have appeared.

One approach is to take an existing language and add to it routines to drive the mechanical devices. This permits the full power of the language to be used. Another is to write library routines, so that the user program consists of a series of calls to these routines in addition to simple control statements. Yet another approach is to design a language specifically for manipulation.

Most of the efforts have been done using the third approach. One of the reasons of this choice is that no available language was at the same time high-level, general purpose, real-time, and supporting cooperant and parallel processes. Moreover an interactive environment for developing programs was available only in LISP-based systems.

The desirable features for a robot programming language have been identified. The language should be general purpose to allow indefinitely complex computations from sensors and vision, should support cooperation to express cooperative simultaneous operations of multiple robots and devices. Since robots work in a real-time environment data must be processed within certain time constraints when they are received, they may be received asynchronously, and the computer should be able to ask for and obtain data at any arbitrary time. The ability to check periodically to see if certain conditions are satisfied in order to synchronize events which are dependent on the conditions or to determine what action to take is another important requirement. Specialized data types should be available to express manipulator positions in the space.

0166-3615/82/0000–0000/$02.75

On the other hand, this great generality may be difficult for the user, so interactive facilities should be available to assist him during the debugging and testing of the application program. For this purpose we have developed the POINTY system, at Stanford Artificial Intelligence Laboratory, as an interactive environment for AL programs.

What will be changing after the availability of ADA? Most of the desirable features are there, so the need of specifically designed languages will decrease. Will ADA be the language for the next generation of robots? We will discuss the strong and the weak points of this choice taking into account both the present needs and the future foreseeable developments of robots.

2. What is a Robot Programming Language?

Since computer controlled manipulators have been introduced as a general purpose mechanism for industrial automation, the methodology of controlling and of programming them for new tasks has seen a great deal of development. Some important issues for these systems have gained a wide acceptance [12].

Robots should be programmed in a simple way, without extensive users training. This goal has been partially achieved with many industrial robots, which are programmed guiding the arm through the motions of the task and storing the sequence of the so obtained positions for further executions.

Teaching by guiding has been successful for tasks where only simple operations or few positions are required. Where complex assemblies are performed that method does not allow any modification or adjustment of the movements during the execution, and makes it impossible to use force sensing and vision. Even small changes in the assembly station cannot be introduced without repeating all the teaching.

On the other hand, writing programs is not so easy as one might think. Manipulation and assembly tasks are difficult to program because the expression of movements in terms of manipulator positions requires many details. The intuitive knowledge about physical operations can be hardly expressed by words.

The increasing interest in the use of sensory feedback, mainly for assembly operations, makes the availability of a programming system for controlling the operations of the robot a real need.

There are two aspects in a robot programming language. The user language, in which application programs are written, and the run time system, which executes the code that results from compilation or interpretation of a program in the user language.

When a robot is running without any response to sensory data a simple run time system can be used, which controls the robot through a fixed sequence of joint positions.

When a sensory response is required, computations of arbitrary complexity are required at run time [15].

Several sorts of response to data obtained from sensors or vision can be envisaged. For instance, in [13] the following are considered:

- a discret choice between one sequence of actions and another is done at run time;
- the values obtained are used to take an action which depends quantitatively upon sensory data;
- the sensory data are used to control the movements of the robot continuously during an action.

The robot programming languages can be conceptually divided into four classes, according to the level in which operations are expressed: joint level, manipulator level, object level, and task level [9,10].

Starting from the lowest level we have *joint level* languages. The description of a task is expressed in terms of the control commands required to drive the individual motors and actuators. Each joint is explicitly controlled. That means that the user should program directly in the joint space instead of in the cartesian space, and should know the law of operation of the motors.

At the *manipulator level* we insert all the languages in which the user controls manipulator positions and movements in the cartesian space. There is no explicit representation of the objects which exist in the world where the robot is. Let us indicate MAL [6] as an example of this class. Other well known examples are the VAL language of Unimation and the RAIL language of Automatix.

On the *object level* we have languages which have some knowledge about the objects. This knowledge is usually partial, because complete object

models are not always needed, but only some features are relevant for the task. Object models are used to describe the sequence of operations with less details or to compute collision free trajectories [3]. In this class we may consider languages as AL [5,8], in which objects are represented through six coordinates as rigid bodies in the space, or RAPT [13], AUTOPASS [10], and LAMA [11], in which incomplete or complete models based on geometry are given.

Most of the research problems still open in robot programming languages are at this level, while the manipulator level is better assessed. We will come back later in this section on the features of this level.

In the *task level* we intend to have those systems which are able to understand and execute descriptions of the task. At this level there are no working systems, although some of the systems illustrated in the previous class are oriented towards the task level.

The four levels can be shortly illustrated by the example given in Fig. 1 where some instructions for the different levels and the conceptual operations which transform each level into the lower are indicated.

Since AL is, at the moment, the most advanced system among those indicated which is completely implemented and running we will refer to it. Although future developments of research will create more advanced languages it is reasonable to consider a language that has been successfully tested by a community of users which is large enough.

We will present conclusions attained though our experience in developing POINTY [2,7,8], the interactive environment of AL. It is interesting to observe that the views expressed by Rieger et al. [14] in their exploratory study do agree with many of our opinions.

The ideal system should contain as desirable features a simple syntax and semantics, to make program readable, and in addition it should take care of keeping the following desirable features.

Data structures should accomodate complex symbolic information as well as primitive types. For instance data types as rotations or frames, which are used to describe object positions, should be available as abstract data types. The user should refer to them instead of their internal representation, which can be realized by orthogonal matrices or quaternions. Strong input/output and file manipulation facilities are highly desirable. It should be possible to save programs and data both in symbolic and internal form; it should be easy to access files from programs, and log files should be provided by the system. A generalized form of input/output instruction should accomodate for data coming or going to different external devices.

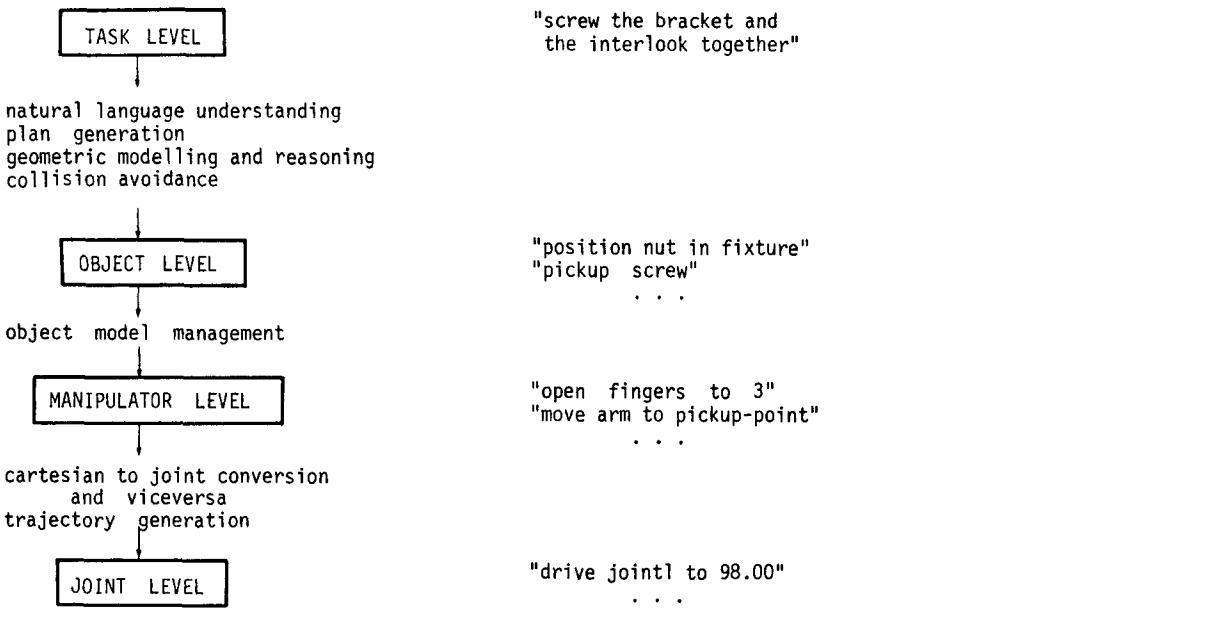


Fig. 1. Levels of Robot Programming Languages.

The control of parallel events should be easily available, both to coordinate different robots and devices and to implement parallel algorithms.

Robots should work in an automated factory and cooperate with other machines, all under computer control. The software should take care of communication in a local area network.

For software development and debugging an interpreter should exist for the language or a reasonably complete programming environment should be provided. Nevertheless, the language may have a compiler for production usage.

A set of utility packages, as a language oriented editor, a debugging module, a source program formatter, a self-modifying display, should be made available.

3. ADA: A Language for Robot Programming

We are now going to investigate the use of ADA [1] as a programming language for robots. As indicated in the previous section, we are mainly concerned with the application of robots to assembly operations, which appears to be a worthwhile domain for programmable automation.

We will not investigate here the use of ADA as implementation language for robotics, both in the sense of language for implementing the run time system and as language for implementing the compiler (or interpreter) of the robot programming language. We may reasonably suppose that ADA can be well suited as implementation language for run time systems.

ADA is a real time and a high level language, qualities which are very difficult to find in the same language and both equally important.

We do not want to enter into the discussion about the performance of ADA as a real time language. It is certainly true that languages which have large overheads tend to be unsuitable for real time programming. Robots are systems whose correct functioning depends on how much time is spent in computation and how fast requests are satisfied once they are sent.

Only the availability of good ADA compilers will allow a more precise evaluation of this aspect.

We will examine whether the features offered by ADA are corresponding to the needs of programming languages for robots. According to the indications given in the previous section, we will consider, in particular, the features offered by AL [5].

One of the positive aspects of ADA is the possibility of using packages. ADA, through packages, provides clean user interfaces to complicated sets of hardware. It is possible to encode the features of the I/O interfaces and external devices, together with their addresses and other characteristics, within package bodies, leaving only those features that are needed by the user in the visible part of the package.

Taking into account the fact that robotic systems heavily depend on the hardware, which is the manipulator as well as the computer, the possibility of hiding the hardware details from the user can be considered as truly important.

As representations in ADA are associated with types rather than variables, different hardware devices require different ADA instructions. Each robot or each device will require a specific package that could be provided together with the device. In this way the hardware aspects are hidden from the user, who can write simple code sequences. For instance the user will be able to write instructions such as

```
SET_DIRECTION (SCREWDRIVER, CLOCKWISE)
```

without worrying about where SCREWDRIVER is in memory or what are the characteristics of the interface.

Another strong point in favor of ADA is the standardization, which will be strictly enforced, and the consequent portability of programs between different systems. Moreover the availability of ADA will increase the effort spent into the development of suitable ADA packages. For the robot producers it will be much easier to develop specific ADA packages for their systems than to develop a specific programming language, as they are doing today. We may mention the VAL language, developed by Unimation for the PUMA system, the SIGLA language of Olivetti for the SIGMA robot, and the language developed by DEA for the PRAGMA assembly system.

The standardization of ADA could enforce a sort of standardization in the programming aspects of robots. That can be considered an important achievement.

The fact that AL is becoming a standard, at least for the research groups in the United States, indicates that the need of a high level language for

robotics is widely recognized. The dissemination of the AL system has been obstacled by the fact that AL requires a large computer (PDP 10) plus a PDP 11. Moreover AL is implemented both in a language available only at the Stanford Artificial Intelligence Laboratory (SAIL) and in the assembler of the PDP 11.

Although AL is obviously more suited to robotics than ADA, the big effort put into the development of the ADA system will eventually enable ADA to become the standard language for robots. It is important to recognize that the main result of the AL project is the demonstration that the features available in general purpose high level languages are important in robot programming, that it is possible to run, efficiently, programs written in a sophisticated language, and that it is much easier to write programs in a high level language than in a joint level or low manipulator level language. Those achievements have been recognized in different meetings and by various people working with the AL language.

Coming back to the features of ADA, we may note that ADA allows the definition of data types specially suited to the needs of manipulation, as the types vector, rotation, transformation, and frame of the AL language. Through overloading the meaningful operations on those data types can be easily defined.

Also in this case the package construct makes the internal representation of data types hidden to the user, so realizing abstract data types. That is particularly important since the internal representation of rotations and frames would be changed within the package body, for instance from orthogonal matrices to quaternions, without affecting user programs.

An example of an ADA package defining the basic data types of AL and the operations on them is illustrated in Fig. 2.

Other features of ADA such as task synchronization and parallelism and exception handling are obviously important for robotics.

It is well known that the handling of errors is one of the most difficult parts in robot programming. Errors are often unpredictable and it would be unfeasible for the user to take explicitely into account all of them. The exception handling mechanism of ADA seems to be the best solution to this problem. The user defines the conditions for rising errors and writes the appropriate recovery procedures, without worrying about their activation.

```
package AL_DATA_TYPE is
  type SCALAR is private;
  type VECTOR is private;
  type ROT is private;
  type TRANS is private;
  type FRAME is private;
  STATION: constant FRAME;
  XHAT, YHAT, ZHAT, NILVECT: constant VECTOR;
  NILROT: constant ROT;
  NILTRANS: constant TRANS;
  function "*" (X, Y: TRANS) return TRANS;
  -- trans composition
  function "*" (X: TRANS; Y: FRAME) return FRAME;
  -- frame transform
  function "*" (X, Y: ROT) return ROT;
  --rotation composition
  function "*" (X: TRANS; Y: VECTOR) return VECTOR;
  -- vector transform
  function "*" (X: ROT; Y: VECTOR) return VECTOR;
  -- vector rotation
  function "*" (X: SCALAR; Y: VECTOR) return VECTOR;
  -- dilation
  ...
private
  type SCALAR is digits 8 range -1E30..1E30;
  type VECTOR is array (1.. 3) of SCALAR;
  type ROT is
    record
    AXIS: VECTOR;
    ANGLE: SCALAR;
    end record;
  type FRAME is
    record
    ROTATION: ROT;
    POSITION: VECTOR;
    end record;
  subtype TRANS is FRAME;
  ZHAT: constant VECTOR;
    NILVECT: constant VECTOR := (0.0,0.0,0.0);
    NILROT: constant ROT := (ZHAT,0.0);
    NILTRANS: constant TRANS := (NILROT, NILVECT);
end AL_DATA_TYPES;
```

Fig. 2. Specifications of AL_DATA_TYPES package.

We do not want to spend more words on that. We prefer to approach other aspects of robot programming the solutions of which are not immediately obtainable in ADA.

4. Future Developments of Robot Programming Systems

As we have indicated in Section 2, robot programming systems can be classified into different

classes. The idea which is at the basis of the classification is to differentiate languages according to the amount of specific knowledge that the user should put into programs.

More knowledge is embedded in the programming system, less detailed will be the specification of the task from the part of the user.

The writing of programs is easier if the language allows to express task oriented specifications more than action oriented specifications. In action oriented programming the description of the actions of the robot is of primary concern, and the description of the objects on which the robot operates is considered as incidental. Alternatively in task oriented programming the description of the task we want to realize and of the objects on which the robot operates is the most important aspect, and the robot actions are considered as a consequence of the description of the task.

The example illustrated in Fig. 1 shows how a program at joint or manipulator level tends to be long and device dependent, while the same program at object or task level is shorter and easier to understand.

The task level description, as indicated before, can be adopted provided that the appropriate general knowledge is given to the system. As indicated in Fig. 1, the transformation of task level descriptions into executable operations requires the system to be able of dealing with natural language, of generating plans for actions, of reasoning about geometry, and of computing collision free paths.

A few systems currently being developed makes use of world models in different ways. One of the most investigated field is how to plan collision-free trajectories. A general solution to this problem, as the one proposed by [3], requires complete body models and operates only for objects with planar surfaces. Another solution, used in AUTOPASS [10], makes use of some heuristic criteria. Neither in RAPT [13] nor in AL [5] this capability is provided.

Generating trajectories can be approached as a special case of geometric reasoning Some systems made recently available, as ACRONYM [4], incorporate a large amount of knowledge about geometry and properties of objects. They can be applied to the computation of collision free trajectories although their main purpose is more general.

Geometric reasoning appear to be the central issue for the next generation of robot programming systems. The ability of dealing with geometrical models of objects is in fact the major requirement for transforming task descriptions into sequences of operations of the robot. The description of the objects could be expressed in natural language or could be directly obtained from a vision system, as done by ACRONYM.

The other capability we mentioned before is the ability to construct plans of actions from very high level descriptions. This research direction, developed within Artificial Intelligence [17], has reached many objectives in the middle of the 70's [16]. The application of plan generation to robots has failed in the past because of the limited capabilities of both planners and robots.

According to Sacerdoti [16], the industrial robots available today are able to perform fixed sequences of actions activated by simple visual or contact stimuli. The sequence of actions is performed without significant alterations whenever the stimulus is presented, no feedback is used when a new stimulus appears. The difference between the behavior of present and future robots is expressed in [16] using the terms of psychology: reflex robot versus instrumental robot.

What will be the role of ADA in those future developments of robot systems?

We may note that most of the problems mentioned above are still in the research stage. The fact is that we do not know yet how to make really intelligent robots and that will be a research topic for next years.

The problem of the choice of the programming language tends to become important when all the aspects of the problem have been clarified and the basic issues individuated. A programming language is only a tool for expressing concepts and, obviously, it cannot solve problems not yet solved.

In this sense we may say that a discussion about the role of ADA in this field is premature, for some aspects, or it should be brought to a more general discussion: would ADA be a good programming language for Artificial Intelligence? This topic would take us very far from the aim of this paper.

5. Concluding Remarks

We have examined the characteristics of the programming languages for robots and we have

proposed a classification of them in four levels, according to the amount of knowledge embedded in the language.

The use of ADA as a language for programming robots has been explored, and we have concluded that ADA can serve as a valuable tool for writing applicative programs at the same level as AL, provided that adequate packages are available for dealing with the hardware interfaces.

In closing, we have indicated the *open* problems for the *next* generation of robots and we have illustrated some of the *functional capabilities* that will be necessary.

Acknowledgements

Partial support for this paper has been provided by the CNR through Progetto Finalizzato per l'Information, Sottoprogetto P3, Obiettivo MODIAC.

References

[1] Reference Manual for the ADA Programming Language. Proposed Standard Document, Dept. of Defense, USA, July 1980.

[2] Binford, T.O., Liu, C.R., Gini, G., Gini, M., Glaser, I., Ishida, T., Mujtaba, M.S., Nakano, E., Nabavi, H., Panofsky, E., Shimano, B.E., Goldman, R., Scheinman, V.D., Schmelling, D. and Gafford, T., Exploratory study of Computer Integrated Assembly Systems, Progress Report 4, SAIL Memo AIM-285.4, Stanford University, Ca, June 1977.

[3] Boyse, J.W., Interference Detection among Solids and Surface, *Comm ACM*, Vol. 22, N. 1, January 1979, pp. 3–9.

[4] Brooks, R.A., Greiner, R. and Binford, T.O., The ACRONYM model-based vision system, Proc. 6th IJCAI, Tokyo, Japan, 1979, pp. 105–113.

[5] Finkel, R., Taylor, R.H., Bolles, R.C., Paul, R. and Feldman, J.A., Overview of AL, a programming system for automation, Proc. 4th IJCAI, Tbilisi, USSR, 1975, pp. 758–765.

[6] Gini, G., Gini, M., Gini, R. and Giuse, D., A multitask system for robot programming, *ACM Sigplan Notices*, Vol. 14, N. 9, May 1979, pp. 11–18.

[7] Gini, G. and Gini, M., POINTY: a philosophy in robot programming, in (Rembold Ed.) *Information Control Problems in Manufacturing Technology*, Pergamon Press, 1980.

[8] Gini, G. and Gini, M., Interactive development of object handling programs, *Computer Languages*, Vol. 7, 1982, pp. 1–10.

[9] Latombe, J.C., Une analyse structureé d'outils de programmation pour la robotique industrielle, in *Langages et methodes de programmation des robots industriels*, IRIA, Paris, France, 1979.

[10] Lieberman, L.I. and Wesley, M.A., AUTOPASS: an automatic programming system for computer controlled mechanical assembly, *IBM Journal of Research and Development*, Vol. 21, N. 4, July 1977, pp. 321–333.

[11] Lozano-Perez, T. and Winston, P.H., LAMA: a language for automatic mechanical assembly, Proc. 5th IJCAI, Cambridge, Mass, August 1977, pp. 710–716.

[12] Nitzan, D. and Rosen, C.A., Programmable industrial automation, *IEEE Trans on Computers*, Vol. C-25, December 1976, pp. 1259–1270.

[13] Popplestone, R.J., Ambler, A.P. and Bellos, I.M., An Interpreter for a language for describing assemblies, *Artificial Intelligence*, 14, 1980, pp. 79–107.

[14] Rieger, C., Rosenberg, J. and Samet, H., Artificial Intelligence Programming Languages for Computer-Aided Manufacturing, *IEEE Trans on Systems Man and Cybernetics*, Vol. SMC-9, N. 4, April 1979, pp. 205–226.

[15] Rosen, C.A. and Nitzen, D., Use of sensors in programmable automation *Computer*, December 1977, pp. 12–23.

[16] Sacerdoti, E.D., Plan generation and execution for robotics, NSF Workshop on robotics research, Newport, Road Island, April 1980.

[17] Winston, P.H., *Artificial Intelligence*, Addison-Wesley Publishing Co., 1977.