AUTOMATIC GENERATION OF ROBOTS PROGRAMMES FROM QUALITATIVE DESCRIPTION OF TASKS

Draghita IANICI
Sava IANICI

Abstract: Automatic generation of programs for robots, starting from a qualitative description of applications, using a quasi-human communication with a media of programs development is tackled. The knowledge base is built on the theory of frames and parametrized objects used for describing both the scene and application.

Key words: robot programming, frames, task planning, qualitative quantities.

1. INTRODUCTION

The majority of industrial robots working in production lines or flexible cells fulfil their task that are fixed for a technological process. The evolution of local environment is generally perfectly predictable. Sometimes sensors supervise some parameters of the environment and determine reactions according to its changes.

The programming of these robots is performed by teaching or using some formal languages (off-line programming). There are much such languages. The most significant are: AML - IBM; VAL - Unimation; RAPT - Poppelstone; AUTOPASS - Lieberman & Wesley et al. Extensions of some of these languages allow the coordination of robot activity with the information taken from visual sensors (VAL, RAPT).

Robots programming requires a qualified staff, which is legitimate only in enough large industry. In small industrial units or workshop it is desirable that the user of a robot should be helped to transmit tasks to the robot in a natural way as simple as possible like in the interhuman relationships.

An intelligent system should be in charge of robot programs generation. There are activities to make robots able to fulfil complex tasks in unpredictable environment. Robots must be able to make reasoning and to adapt their behaviour relying on the information taken from sensors. Merging achievements in the artificial intelligence with robust techniques in robots control and trajectory planning a dedicated tool may be Knowledge representation is made by frames. Objects, actions, scenes activities are represented by frames. Objects are actions constitute a library of parametrized primitives, an a priori knowledge. The access to the knowledge elements is made by a restriction of the English language. The system allows the description of the operational world, task planning, trajectory generation and means of simulation during development.

At the robots motion level the resulting program, expressed in an operation language, is translated by a post processor into the robot’s off-line language.

2. THE SYSTEM ARCHITECTURE

The development of such a complex system must be approached at different levels of abstraction. The architecture is defined at the highest level.

The proposed architecture is based on frames for representing the model of the environment, as well as for describing the actions required for the execution of a specific task. A frame [1] is a data structure that consists of a network of nodes and relations. A stereotype situation or action is represented by a frame. The structure of a frame as \( P = \{E, S, F\} \), where:

- \( E \) – label, a linguistic value to which usually general and static information is associated (always true) describing the entity represented by the frame;
- \( S \) – slots (terminals), in which specific and dynamic information is stored;
- \( F \) – facets, each terminal may have a set of facets with corresponding values or a set of subframes (references to other frames).

A prototype frame represents a class from which instance frames are derived. The syntactic structure based on frames defines semantic connections for particular situations.

The slots are locations for semantic primitives so that the knowledge element is placed in the frame context. One of the essential of this knowledge representation is that on the basis of some slots available in an exemplar frame and corresponding slots coming from the prototype frame the slots with missing value can be filled in. The slots containing values are viewed as qualitative quantity spaces or fuzzy sets covering finite universes.

The use of a formal system based on well-formed formulas and predicate calculus for constructing plans is not appropriate in representing low-level actions like motor and sensing functions which are robot-specific. Frames are well suited for representing the high-level symbolic constructs and the lower-level knowledge
procedures and constraints, [2]. Frame representation of planes allows hierarchical and iterative planning. At the action level a functional programming is structured. Procedural knowledge and operations can be easily encoded in structured representations. The basic modules of the system and their main functions are:

- **User Interface**
  - Commands Acceptance
  - Manual Mode
  - System Status

- **Supervisor**
  - Event Scheduling
  - Event Monitoring
  - Interrupt Handling

- **Task Planner**
  - Plan Identification
  - Plan Acquisition
  - Plan Decomposition
  - Plan Instantiation
  - Plan Generation

- **Knowledge Base**
  - Long Term Memory
  - Short Term Memory
  - Data Acquisition and Maintenance

- **Trajectory Generator Subsystem**
  - End - Effector Motion
  - Tool Manipulation
  - Control Signal
  - Sensor Placement

- **Perception Subsystem**
  - Data Preprocessing
  - Data Interpretation
  - Sensor - World Transformation

- **Graphical Simulation Subsystem**
  - CAD-Tools for Wire-frame Modelling
  - Object Library
  - Topological Facilities

These modules are structured as in figure 1 and the functional structures depicted in figure 2.

![Fig. 1. System architecture](image)

![Fig. 2. Functional structure of the system](image)

## 3. AUTOMATIC TASK PLANNING

Automatic Task Planning is a major area of research in the artificial intelligence science. The Task Planning in robotics can be defined as the generation of a set of actions that enable the system to achieve a goal by changing robots work environment. Two basic approaches of planning can be considered:

- The first approach is the state-space approach where a path from an initial state to the final state is searched. The state-space is defined as:

\[
S = < s, \Phi >
\]

where: \( s = \{s_i\} \), \( s_i \) are distinct states of the environment;

\( \Phi = \{\phi_i\} \), \( \phi_i \) are operators, partial mapping of states into states set.

A problem \(< s_0, G >\) in \( S \) is the finite sequence \((\phi_1, \phi_2, ..., \phi_n)\) of operators for which there is a sequence of chained transitions:

\[
s_0 \xrightarrow{\phi_1} s_1 \xrightarrow{\phi_2} ... \xrightarrow{\phi_n} s_n
\]

where:

\[ s_n \equiv G \]

One can immediately see that in this setting the task of plan formation is equivalent to that of problem solving.

- The second approach is action-ordering approach where given an initial high-level plan (in a sketch form), a set of detailed plans leading to the completion of the goal is generated, [2]. STRIPS was a planner for solving the classic “blocks world” problem.

In a static environment, starting from an initial state, using a set of operators able to change the state, a string of operators is found capable to lead to goal accomplishment. The plan of a task should be described at various levels of details and abstraction.

One can define the following notions:

- plan: is an ordered sequence of operators or plans.
  - There are two kinds of plans: primitive and complex.
- primitive plan: is that one that can be directly excluded. A primitive plan contains only one operator instantiated by parameters values.
complex plan: cannot be directly executed and are user-specified or automatically generated by the Task Planner. The predefined plans that are acquired through the Knowledge Acquisition Module and stored in the Knowledge Base are fetched by the Task Planner. This way the user may communicate the task assignment at the highest level to the intelligent system. A hierarchy of subplans with details required for the accomplishment of the overall task associated to the higher level plan. At the lowest level of this hierarchy are the primitive plans for trajectory generation. The use of frames allows a good structure for this hierarchy. The installation of parametrized primitives is performed by values assignment to the parameters. The planning is bound to the user interface that is enabled to structure an initial plan that is refined and detailed by the system in cooperation with the user included in that loop. In order to accomplish a complex goal, the robotic system will require specification of a very detailed sequence of individual action tasks. The system must have the ability to describe plans at various levels of details and abstraction. The user communicates the task assignment at a high level. Associated with a much higher level plan there will be a hierarchy of subplans with increasing amounts of specific details. At the lowest level of this hierarchy are executable primitive plans. For practical systems, one needs to provide a capability to interrupt a normal course of operation of the system in the event of a higher priority task assignment. A sequence of available operations for the service of the interrupt is generated.

The plan generation is accomplished as follows:

- the predefined plans stored in Knowledge Base are fetched by the Task Planner. This occurs if suitable plans exist;
- if the fetched plan is a primitive one it is instantiated at the time the plan is fetched.

Instantiation is the assignment of specific values to the input parameters. For a movement plan, instantiation includes specifying the motion velocity, destination position and orientation, the reference system name and the parent plan name.

The plane generation by the Task Planer is described in the next algorithm:

A. Initialization
B. Supervisor Module:
   - Acquire Top-Plan
C. Task Planner:
   - Generate/Fetch Complex Plan
D. Fetch Sub-Plan
E. If Plan Type Complex then Go to C else (Primitive)
F. Task Planner:
   - Fetch Primitive Plan
G. Supervisor:
   - Activate Trajectory Generator
H. Trajectory Generator
I. Success, Failure, Interrupt Servers
J. If No More Sub-Plan then Exit else
K. Determine Next Sub-plan, Go to D

Plains can be constructed by the user through User-Interface. This is a learning by example. By means of primitive plans a complex plan can be constructed. This plan characterized by some features can be saved in the Knowledge Base so that the next time a similar situation happens the plan can be fetched and applied.

4. THE QUALITATIVE REPRESENTATION AND LINGUISTIC EXPRESSIONS

The inferences necessary for robot’s motion planning can be developed profitably by qualitative representation of spatial relationship between object and of the robot itself. The relations between objects can be represented by a formalism of qualitative measurements. The qualitative spatial quantities (variables) of interest are Euclidean distances, linear displacements, angular displacements, relative positions and orientation and location, located in slots of particular frames.

Firstly a no nonambiguos vocabulary is chosen which allows efficient and nonambiguous spatial inferences. A qualitative quantity space is defined as it follows, [3]. Let x be a quantitative variable, such that \( x \in R \) and \( R \subseteq R \). If the entire domain \( R \) partitioned into finite set of mutually disjoint subdomains:

\[
\{ Q_1, Q_2, \ldots, Q_n \}
\]

i.e.:

\[
\bigcup_{i=1}^{n} Q_i = R
\]

and all numerical values lying within \( Q_i \) are treated as being equivalent and named symbolically under Label \( (Q) \), then the qualitative variable \( [x] \) corresponding to \( x \) is defined as follows:

\[
[x] \in X, X \subseteq \bigcup_{i=1}^{n} Label(Q_i)
\]

where \( Label(Q) \) is called primitive qualitative value. The union \( X \) is called the qualitative quantity space of \( [x] \), denoted by \( Q - space_{[x]} \). Labels correspond to qualitative distances. Usually the distance is expressed relative to another reference distance. The primitive values (labels) of a qualitative distance, \([x/d] \), are defined as following:

\[
less \Delta / x / x \in R, x/d \in (0,1/2) \}
slightlyless \Delta / x / x \in R, x/d \in (1/2, 1) \}
equal \Delta / x / x \in R, x/d = 1 \}
slightlygreater \Delta / x / x \in R, x/d \in (1, 2) \}
greater \Delta / x / x \in R, x/d \in (2, \infty) \}
\]

Given two distance variables A and B the following relations are valid:

\[
[A/B] = less \Leftrightarrow [B/A] = greater
\]

In an analogous mode a qualitative space for angles may be constructed. Tabular matrices are built based on qualitative relations on which inferences may be performed. The entries of the slots of given frame can be specified as well as fuzzy sets referred by linguistic values too.
The spatial inferencing is concerned with the qualitative spatial analysis of the relative distances and angles of orientation.

## 5. CONCLUSION

An environment meant to serve in robot’s planning and programming may be considered the solution to facilitate the user’s access to robotics facilities. This tool can be considered formed by a profession oriented expert system that allows the instantiation of some primitive or complex objects and actions from a knowledge base for work environment and tasks definitions. The robot’s program related to environment is generated. Linguistic descriptions allow the initial qualitative definition of the environment, of the objects, activities and of the robots itself (its definition is fixed, referred by its name for all applications). Prototype frames are related to linguistic values. For some objects and actions quantitative statements must be filled in appropriate slots. At the qualitative level the first approximation is made for trajectory generation using searching and qualitative reasoning. The very trajectory generation is quantitatively performed on the qualitative path previously obtained. The use of sensors, in the meaning of verification vision, [4], enables the adjustment of the generated programs for “normal” situation, to the alterations of position and orientation of objects in the work environment. This alteration must be in an accepted boundary. Another part of this system enables the user to simulate the task performing execution. The analysis of simulation allows program alteration to improve it and its behaviour based on user intervention. Such programming environment is in the stage of development under project name PROROB.

## REFERENCES

### Books


### Journal articles


## CORRESPONDENCE

Draghiţa IANICI, As. drd. eng.
Eftimie Murgu University of Reşiţa
Faculty of Engineering
Traian Vuia Square 1-4
320085 Reşiţa, Romania,
d.ianici@uem.ro

Sava IANICI, Prof. dr. eng.
Eftimie Murgu University of Reşiţa
Faculty of Engineering
Traian Vuia Square 1-4
320085 Reşiţa, Romania,
s.ianici@uem.ro / decaning@uem.ro