TOWARDS AUTOMATIC ERROR RECOVERY IN ROBOT PROGRAMS

Maria Gini
Department of Computer Science
University of Minnesota
207 Church St. SE, Minneapolis, MN 55455

Giuseppina Gini
Dipartimento di Elettromca
Politecnico di Milano
piazza L. da Vinci *32,* 20133 Milano. Italy

ABSTRACT

Unexpected events can cause the failure of apparently "correct" robot programs. The interaction with the real world and its unpredictability make the problem of error recovery in robot programming specially important. The goal of the paper is to present a general framework in which the activity of error recovery can be automated. This is accomplished by introducing a monitor program which identifies the appearance of any error and attempt to correct that error. The correction is done using a knowledge base where the knowledge that the user has about error identification and correction is expressed in symbolic form. An inference mechanism allows extension of this knowledge base for use in complex and unanticipated situations.

I INTRODUCTION

Robots are being used in a wide variety of applications. To operate successfully they should be able to handle unexpected events. A more intelligent perception of the robot environment is needed The capability of making decisions in answer to external conditions should be improved. This should also result in greater safety for the operating personnel and the equipment installed in the vicinity of the robot

With current robot programming languages (Bonner, 1962), one can recover from failures caused by arm errors only by using ad hoc error recovery procedures. In writing and debugging manipulation programs, users must depend on their experience, intuition, and common sense to decide what errors to watch for.

Errors in robot programs are difficult to identify because of their unpredictability. The same program can work well hundred of times and then stop because of a minimal variation in size of one part or because of a little spot of oil on it. Moreover since the programming is done on-line (Gini, 1982) the robot must be used for large amounts of time to check new programs before they can be reasonably used in production.

The problem of recovering after an error has not yet been fully addressed. To do this the system needs to have a knowledge of how the world in which the robot is operating is structured (Gini, 1981).

The problem of dealing with errors has been approached in various ways and with different objectives in plan generation research. Systems such as NOAH (Sacerdoti, 1977), and HACKER (Sussman, 1975). tried to solve errors arising during the planning. The TROPIC system (Latombc, 1979) has a similar mechanism for failure correction. These approaches have not been applied to real robot tasks.

Partial support for this work is gratefully acknowledged to the Microelectronic and Information Sciences Center at the University of Minnesota and to the Italian National Council of Research.

The system more close to our solution is presented in (Srinivas, 1976). He has designed a practical system for analysing failures and their causes, and for replanning the recovery activity. Its main limitation derives from the extensive use of plan formation as the basis for constructing robot programs and on the choice of checking only the preconditions of the actions. In this way an error may be discovered later than when it appeared.

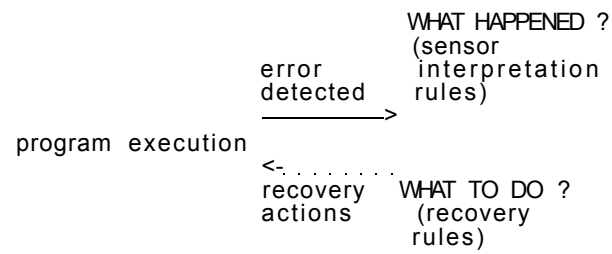
The problem of error recovery plays an important role in industrial robotics The possibility of using robots unattended, such as during the night, requires at least a reasonable solution of the problem. Strategies to fulfill safety requirements in the case of failures of the robot are important too.

// A METHOD FOR AUTOMATIC ERROR RECOVERY

This paper presents a general framework for automating the error recovery activity This is accomplished by an intelligent monitoring system running concurrently with the robot program Every time an error arises the appropriate recovery procedure is detected using information extracted from a knowledge base (Stefik, 1982), The knowledge base contains rules about correction activities and about interpretation of sensor data.

To detect what happened and to identify the recovery action the system .should know the effect on the world of each of the instructions of the program. Some form of dynamic model of the robot environment and the ability of interpreting information gathered by sensors are also needed (Rosen, 1977)

The general scheme is



We examine in more detail the recovery method. We start by defining the dynamic model of the world, and the semantics of the robot programming language. Then we present the organization of the knowledge base.

A. Dynamic model

An initial model of the world is constructed from the declarations present in the program, and data from sensors.

For each instruction let InitialModel be the model valid before the execution of the instruction.

ExpectedModel is the model expected to be valid after the execution of the same instruction It is obtained from the InitialModel and the postconditions of the instruction.

ExpectedModel can contain conditional expressions since postconditions can be expressed with conditional parts. For instance the instruction that, closes the hand can be used either to grab an object or to close the hand. The sensors in the finger can identify the situation at run time

Let CurrentModel be the currently valid model.

It should be obvious that if there are no errors before executing an instruction CurrentModel is the same as InitialModel If there are no errors after executing the instruction CurrentModel is the same as ExpectedModel.

1. Example

At the beginning of the program the InitialModel can be:

```
Arm = ParkPosition
HandOpening = X
if TouchSensorTriggered
   then ObjectHeld,
        ObjectSize = HandOpening,
        ObjectPickedUpAt = Arm
   else ClearHand.
```

B. Semantics

We describe the semantics of the language in a STRIPS-like form (Fikes, 1971).

Each instruction has associated a list of preconditions and postconditions. The preconditions express what should be true before executing the instruction, the postconditions express how to modify the current model after the execution of the action. They are expressed in term of additions (ADD), deletions (DEL) , and updating (UPD) to the model.

1. Examples

We consider a small subset of AI. instructions (Binford, 1979).

```
MOVE ARM TO frame
prec:
post: UPD: arm=frame,

OPEN HAND TO d
prec:
post: if ObjectHeld
         then ADD: ClearHand
              DEL: ObjectHeld,
                   ObjectSize = X.
                   ObjectPickedUpAt = Y
      UPD: Opening = d

CLOSE HAND TO d
prec: ClearHand
post: UPD: HandOpening = d
      If TouchSensorTriggered
         then ADD: ObjectHeld.
                   ObjectSize= d,
                   ObjectPickedUpAt = Arm
              DEL: ClearHand
```

Note that we consider rigid objects so that after OPEN and before CLOSE the hand does not hold anything.

Using postconditions the ExpectedModel and the CurrentModel can be determined For instance, after a MOVE instruction the ExpectedModel is computed by updating the arm position in the InitialModel, while the CurrentModel is computed by reading the actual arm position.

C. Knowledge base

We use a knowledge base containing two types of rules, sensor rules (used to interpret, the sensor data), and recovery rules (used to produce the recovery)

Sensor rules have the form

if D, . then C

where the D's express what we want to know from sensors and C is their "logical" interpretation This organization allows a certain independence between the raw data from sensors and their interpretation

The recovery rules have the form

to obtain G, .. when S, . do R, ..

where the G's express what we want to achieve, the S's express what we know is true, and the R's are recovery actions.

1. Examples

SensorRules:

```
if FingerTouchSensorTriggered
   then ObjectHeld

if not FingerTouchSensorTriggered
   then ClearHand
```

RecoveryRules:

If the object is lost during the movement we can recover with

```
to obtain ObjectHeld
   when   ClearHand
   do     Compute NextPickUp;
          GrabObject(NextPickUp,ObjectSize)
```

knowing that

```
if ObjectPickedUpAt = X
   then NextPickUp = X + d
```

If the arm is not in the right place we can use the rules

```
to obtain  Arm=Frame2
   when    Arm=Framel,  Di st (Arm, Frame2)< 5
   do      MOVE ARM TO frame2 DIRECTLY

to obtain  Arrn=Frame2
   when    Arm=Framel,  Di st ( Arm, Frame2)> . 5
   do      MOVE ARM TO frame2
```

If the hand is too closed

```
to obtain HandOpening = ObjectSize
   when   HandOpening < ObjectSize
   do     OPEN HAND TO ObjectSize
```

2 Recovery procedure

The recovery procedure is activated by the identification of an error. As we said before, an error is identified every time CurrentModel at the end of the execution of any instruction is different from ExpectedModel Knowing the situation in which we are and where we want to be the appropriate error recovery rules can be fired.

We control both the preconditions before executing any instruction and the postconditions at the end. The first check should not be needed since we assume that the program does not have logic errors We consider it useful as a protective measure

After the recovery we resume the execution of the original program at the point where it was suspended, The problem of deciding whether to restart it at a different point has not yet been approached

111 CONCLUDING REMAKKS

Although the examples shown are limited ve think to have supported our claim that we have presented a general framework for error recovery in robot programs Research is under way to write more rules, to introduce strategies in recovery, and to extend our work to complete programming languages. A preliminary mplementation is under development

In our opinion the strong points of our method are:

* It is based on the use of a real robot programming language, not a planning system intended for purposes other than manipulator control;
* The reasoning process used in error recovery is based on information provided by sensors. Any sensor can be incorporated, provided that interpretation rules are available;
* The knowledge base can be easily extended lo cover more errors and more recovery procedures;
* The language used to program the robot could be changed, provided that its semantics is supplied in the same form;
* It can be used to recover errors not only for robots but also for more complex automation systems,

REFERENCES

Binford, T., "The AL language for intelligent robot." in *Languages et Methodes de programmation* des *robots industriels.*]R1A Press, Paris, France: IRIA Press, 1979.

2. Bonner, S., and Shin, K, "A comparative study of robot languages." *Computer Magaz 15*:12 (1882) 82-96.
3. Fikes, R.E., and Nilsson, N.J.. "STRIPS: a new approach to the application of theorem proving to problem solving ." *Artificial Intelligence* 2 (1971) 189-208.
4. Gini, G., Gini, M., Somalvico, M., "Deterministic and non deterministic robot programming ." *(Cybernetics and Systems* 12 (1981) 345-362.
5. Gini, G., and Gini, M., "Interactive development of object handling programs." *Computer Languages* 7:1 (1982) 1-10.
6. Latombe, J.C., "Failure processing in a system for designing complex assemblies " In *I\oc. 6th IJCAI. Tokyo,* Japan, August 1979, pp 508-513.
7. Rosen, C.A., and Nitzan, 1)., "Use of sensors in programmable automation." *Computer Magaz.* 10:12 (1977) 12-23,
8. Sacerdoti, E. *"A structure for plans and behavior."* American Elsevier Publ. Company, 1977.
9. Srinivas, S., "Error recovery in a robot system." PhD Thesis, CIT. 1976.
10. Stefik, M., et al , "The organization of expert systems: a tutorial." *Artificial. Intelligence* IB (1982) 135-173
11. Sussman, G. J., *"A computer model of skill acquisition."* American Elsevier Publ Company, 1975