GENERATING PERCEPTION REQUESTS AND EXPECTATIONS TO VERIFY THE EXECUTION OF PLANS

Richard J. Doyle
David J. Atkinson
Rajkumar S. Doshi

Jet Propulsion Laboratory
4800 Oak Grove Drive, Pasadena, CA 91109

ABSTRACT

This paper addresses the problem of verifying plan execution. An implemented computer program which is part of the execution monitoring process for an experimental robot system is described. The program analyzes a plan and automatically inserts appropriate perception requests into the plan and generates anticipated sensor values. Real-time confirmation of these expectations implies successful plan execution. The implemented plan verification strategy and knowledge representation are described. Several issues and extensions of the method are discussed, including a language for plan verification, heuristics for constraining plan verification, and methods for analyzing plans at multiple levels of abstraction to determine context-dependent verification strategies.

1. THE PROBLEM

In a partially-modelled real world, an agent executing a plan may not actually achieve desired goals. Failures in the execution of plans are always possible because of the difficulty in eliminating uncertainty in world models and of a priori determining all possible interventions. Given these potential failures, the expected effects of actions in a plan must be verified at execution time.

In this paper, we address the problem of providing an execution monitoring system with the information it needs to verify the execution of a plan in real time. Our solution is to identify acquirable **perceptions** which serve as more reliable verifications of the successful execution of actions in a plan than do the inferences directly derivable from the plan itself. Assertions which appear as preconditions and postconditions in plan actions are mapped to appropriate sensor requests and expectations describing a set of values. Observing a value from the expectation set on the indicated sensor at the appropriate time during execution of the action implies that the assertion holds. The strategies for verifying the execution of actions are derived from the intentions behind their use. The knowledge of which perceptions and expectations are appropriate for which actions is represented by **verification operators**.

These ideas have been implemented in a working computer program called **GRIPE** (Generator of **R**equests **I**nvolving **P**erceptions, and **E**xpectations). After describing this program, we propose several generalizations of its results by examining the issues involved in **Selection**, or determining which actions in a plan to monitor, and **Generation**, or how to verify the successful execution of particular actions.

1.1. CONTEXT OF THE PROBLEM

Generating perception requests and expectations to verify the execution of actions in a plan is only one aspect of a robust control system for an intelligent agent. Research on such a control system is underway at the Jet Propulsion Laboratory. In our system, known as **PEER** (**P**lanning, **E**xecution monitoring, **E**rror interpretation and **R**ecovery) [Atkinson, 1986] [Friedman, 1983] [Porta, 1986] several cooperating knowledge-based modules communicate through a blackboard [James, 1985] in order to generate plans, monitor those plans, interpret errors in their execution, and attempt to recover from those errors. In this paper we concentrate on part of the execution monitoring task.

The primary application for PEER is the proposed NASA/JPL Telerobot, intended for satellite servicing at the U.S. space station. The current testbed scenarios for the Telerobot include a subset of the Solar Max satellite repairs recently accomplished by shuttle astronauts.

Broadly speaking, the tasks which must be accomplished by execution monitoring are **Selection**, **Generation**, **Detection/Comparison**, and **Interpretation**. The **Selection** task must determine which effects of actions in the plan require monitoring. The **Generation** task involves determining the appropriate sensors to employ to verify assertions, and the nominal sensors values to expect. This is the task accomplished by the GRIPE system, which we discuss in detail below. The **Detection/Comparison** monitoring task handles the job of recognizing significant events on sensors and then comparing these events with the corresponding expectations. Finally, the **Interpretation** task involves explicating the effects of failed expectations on subsequent plan actions. We will discuss all of these in more detail.

1.2. OTHER WORK

Monitoring task execution and feedback have been the topic of research in AI for quite some time. Attention has been focused on monitoring at the task level, the geometric and physical levels, and also at the servo level. Early work which exposed the role of uncertainty in planning and other problems in error recovery includes [Fikes, 1972], [Munson, 1972], and [Chien, 1975]. Sacerdoti discussed the issues of monitoring and verification in NOAH in detail [Sacerdoti, 1974] [Sacerdoti, 1977]. This work illustrated the role which planning at multiple levels of abstraction could play in monitoring. NOAH used the plan hierarchy as a guide for asking a human to verify plan assertions.

Some recent research in planning and execution monitoring has focused on handling uncertainty. A planner implemented by Brooks reasons about the propagation and accumulation of errors [Brooks, 1982]. It modifies a plan by inserting sensing operations and constraints which ensure that the plan does not become untenable. Erdmann's method for planning trajectory motions utilizes a backprojection algorithm that geometrically captures the uncertainty in motion [Erdmann, 1985]. Donald also addressed the the problem of motion planning with uncertainty in sensing, control and the geometric models of the robot and its environment [Donald, 1986]. He proposed a formal framework for error detection and recovery.

[Wilkins, 1982] and [Wilkins, 1985] deals extensively with planning actions to achieve goals. Wilkins also deals with Error Recovery which is the problem of recovering from errors that could occur at execution time. To be precise, Wilkins does not deal with the problem of planning to monitor the plan generated by the planner.

[Tate, 1984] discusses the usefulness of the intent and the rich represtation of plans. He also mentions the issues in goal ordering, goal interaction, planning with time, cost and resource limitations, and interfacing the planner with other subsystems. He also throws some light on some solutions to the problem of Error Recovery. He mentions the problem of Execution Monitoring but has not discussed the problems or issues or any related solutions.

Other recent research has also addressed the problem of using sensors to verify plan execution. Van Baalen [VanBaalen, 1984] implemented a planner that inserts sensory action requests into a plan if an assertion of an operator is manually tagged "MAYBE".

Miller [Miller 1985] includes continuous monitoring and monitoring functions in a route navigation planner. His focus is on the problem of coordination of multiple time dependent tasks in a well-known environment, including sensor and effector tasks involving feedback.

Gini [Gini et al., 1985] have developed a method which uses the intent of a robot plan to determine what sensor conditions to check for at various points in the plan. In the final executable plan, the system inserts instructions after each operator to check approriate sensors for all possible execution errors.

Fox and Smith [Fox et al., 1984] have also acknowledged the need to detect and react to unexpected events in the domain of job shop scheduling.

2. IMPLEMENTATION

The algorithms presented in this paper have been implemented in a working computer program called GRIPE. In addition to the knowledge sources supplied to the program, the basic input is a plan specification as described below. GRIPE's output consists of a modified plan which includes sensing operations, expectations about sensor values to be used by a sensor monitoring program, and subgoals for the planner to plan required sensor operations or establish preconditions for sensing. GRIPE has been tested on a segment of the JPL TeleRobot demonstration scenario and generates a plan of 67 steps modified to include perception requests and additional output, as described above. The examples shown below are drawn from this test case. GRIPE has been tested on examples from the Solar Max satellite repair domain. The system generates a modified plan which include perception requests, as well as expectations about those perceptions and subgoals for acquiring those perceptions. The following sections describe this process in more detail.

2.1. VERIFICATION STRATEGY

The basic input to the GRIPE system is a plan specification. GRIPE prepares a plan for execution monitoring by examining the preconditions and postconditions of each action in the plan. For each of these assertions GRIPE generates an appropriate perception request and an expectation which, if verified, implies that the assertion holds. During execution, an action is commanded when all of its preconditions are verified and its successful execution is signalled when all of its postconditions are verified.

GRIPE prepares a plan for execution monitoring by examining the preconditions and postconditions of each action in the plan. For each of these assertions GRIPE generates an appropriate perception request and an expectation which, if verified, implies that the assertion holds. An action is commanded when all of its preconditions are verified and its successful execution is signalled when all of its postconditions are verified. GRIPE uses dependency information between conditions established as postconditions in one action and required as preconditions in another. If the establishment and use of a condition occurs in consecutive actions, the condition is verified only once. Otherwise, the condition is verified when it is established, and re-verified when it is needed.

The method of verifying the execution of an action is derived by examining the intention behind its use. The knowledge of which perceptions and expectations are appropriate for which actions is encoded in **verification operators**, described below. In the prototype GRIPE implementation, we assume that actions have a single intention. In general, however, the intent of an action may vary according to the context in which it appears.

As an example, consider moving a robot's arm as a precondition to a grasp. This operation may require high accuracy. A combination of sensors such as position encoders in the arm, proximity sensors at the end effector and vision could be used to ensure that the end-effector is properly placed to grasp this object. On the other hand, moving the arm away from the object after the release operation may require very little verification. The available latitude in the final position of the arm may be large. In this case, a cursory check on the position encoders may suffice. In the prototype GRIPE implementation, we assume that actions have a single intention.

2.2. REPRESENTATION

Before examining in detail how GRIPE generates perception requests and expectations, we describe our representation for plans and our models for actions and sensors in the JPL TeleRobot domain.

A plan is a totally ordered sequence of actions representing a schedule of commands to an agent. The dependencies among actions are maintained explicitly.

Actions are modelled in the situation calculus style [Fikes, 1971], with specified preconditions and postconditions. In addition, an explicit duration for each action is determined and represented by a start and stop time. Currently, we model all actions as having the same duration. As an example, the action operator for GRASP is shown in Figure #1.

```
(create-action-operator
:type GRASP
:action (GRASP End-Effector Object before after)
:preconditions
   ((MODEL= (POSITION-OF-MODELLED Object before)
       (POSITION-OF-MODELLED End-Effector before))
    (MODEL= (FORCE-OF-MODELLED End-Effector before)
            0)
    (MODEL= (WIDTH-OF-MODELLED End-Effector before)
           'Open))
:postconditions
   ((MODEL= (FORCE-OF-MODELLED End-Effector after)
       (COMPLIANT-GRASP-FORCE-FOR Object after))
    (MODEL= (WIDTH-OF-MODELLED End-Effector after)
       (GRASP-WIDTH-FOR Object after))))
```

Figure 1: GRASP Action Operator

There are four types of sensors currently modelled for the TeleRobot: position encoders for the arms, force sensors at the end-effectors, configuration encoders for the end-effectors which tell how wide the grippers are held, and vision system cameras. The table shown in Figure #2 associates with each sensor type the actual perception request which GRIPE grafts into plans. We assume that all the sensors except vision can be read passively at any time. The TeleRobot vision system uses CAD/CAM-type models and requires an expected position and orientation to effectively acquire objects.

Sensor	Perception Request
Arm-Kinesthetic	(WHERE End-Effector when)
Hand-Kinesthetic	(CONFIGURATION End-Effector when)
Force	(FEEL End-Effector when)
Vision	(SEE Object Position when)

Figure 2: Perception Requests for Sensors

Note that there are two equivalence predictes, **MODEL=** and **SENSE=**. The **MODEL=** predicate appears in the preconditions and postconditions of action operators and represents a comparison between two assertions in the world model. All reasoning done during planning occurs within the world model. Assertions in the world model are identified by the suffix **-MODELLED.**

The **SENSE=** predicate, appearing in expectations generated by GRIPE, represents comparisons between a perception and an assertion in the world model. These comparisons are the essence of verification. Perceptions are identified by the suffix **-SENSED.**

2.3. VERIFICATION OPERATORS

The knowledge of how to verify the assertions which appear as preconditions and postconditions in actions is captured by **verification operators.** Verification operators map assertions to appropriate perception requests, expectations, and possibly subgoals. The definition of verification operators is shown in Figure #3. As an example, the verification operator for determining that an object is at a particular location is shown in Figure #4.

```
(define-verification-operator
   ;Assertion to be verified.
 assertion
   ;Actions partially verified by this operator.
 actions
   ;Constraints on assertion.
 constraints
   ;The sensor to be used.
 sensor
   ;Perception request which can verify assertion.
 perception
   ;Preconditions for obtaining the perception.
 preconditions
   ;Sensor value which verifies assertion.
 expectation)
```

Figure 3: Verification Operator Definition

Each verification operator is relevant to a single assertion which may appear as a precondition or postcondition in several different actions. Verification operators are indexed under the actions which they help to verify. In our example, there are three steps involved in determining the relevance of the verification onerator shown in Figure #4 to preconditions of the GRASP action shown in Figure #1.

First, the relevant verification operators for the GRASP action are retrieved. Next, an attempt is made to unify the precondition against the assertion pattern specified in each retrieved verification operator. Finally, any constraints specified in the verification operator are checked. These constraints constitute a weak context mechanism; in our example, the specified constraint distinguishes the use of position encoders to verify the location of an end-effector from the use of the vision system to verify the location of an external object.

Once the relevant verification operator has been identified, a perception request and expectation for verifying that the precondition holds at execution time are generated from the appropriate fields of the verification operator. This information is then passed to the real-time execution monitor.

Perception requests are themselves actions to acquire perceptions via various sensors. The use of sensors may also be subject to the establishment of preconditions. In our example, the simulated vision system can acquire an object only if there are unobstructed views from the cameras to the object. Currently, the other three sensors we simulate are passive and do not have preconditions on their use. In this case, GRIPE generates and submits subgoals generated by particular perception

```
  (create-verification-operator
 :sensor
     VISION
 :actions
     (GRASP RELEASE)
 :assertion
     (MODEL= (POSITION-OF-MODELLED Object Moment)
             Position)
 :constraints
     ((NOT (MEMQ Object
           '(Left-End-Effector Right-End-Effector))))
 :perception
     (SEE Object Position Moment)
 :preconditions
     ((UNOBSTRUCTED-PATH
          (POSITION-OF-MODELLED
                'Left-Camera Moment)
              Position
              Moment)
          (UNOBSTRUCTED-PATH
              (POSITION-OF-MODELLED
                 'Right-Camera Moment)
              Position
              Moment))
:expectation
    (SENSE= (POSITION-OF-SENSED Object Moment)
            Position))
```

Figure 4: Example Verification Operator

requests to the planner. The task of the planner is to further modify the plan, which now includes perception requests, so that preconditions on the use of sensors are properly established. This process details the extent of the interaction of monitoring and planning and suggests the issue of how closely the two processes should be interleaved, a problem which has not yet received much close attention.

3. AN EXAMPLE

The following example is drawn from a satellite repair scenario for the JPL Telerobot, described above. Part of the servicing sequence in the previous example involves grasping the handle of a hinged panel on the satellite. A segment of this plan is shown in Figure #5 When GRIPE processes this plan segment for execution monitoring, it inserts appropriate perception requests into the plan and generates expectations about nominal sensor values. The plans given to GRIPE have been hand-generated.

Perform the action
```
    (MOVE right-end-effector handle
          (NEAR (POSITION-OF-MODELLED handle 2))
          (POSITION-OF-MODELLED handle 3)
          2 3).
```
Perform the action
```
    (GRASP right-end-effector handle 3 4).
```

Figure 5: Example plan input to GRIPE

Verify and do the action (MOVE ... 2 3)
using the **ARM-KINESTHETIC** *sensor*
```
    (WHERE right-end-effeetor 2).
    (SENSE= (POSITION-OF-SENSED right-end-effector 2)
            (NEAR (POSITION-OF-MODELLED handle 2)))
    (MOVE ... 2 3)
    (WHERE right-end-effector 3)
    (SENSE= (POSITION-OF-SENSED  right-end-effector 3)
            (POSITION-OF-MODELLED handle 3))
```

Verify and do the action (GRASP ... 3 4)
using **VISION** *sensor, the* **FORCE** *sensor,*
and the **HAND-KINESTHETIC** *sensor*
```
    (SEE handle
       (POSITION-OF-MODELLED right-end-effector 3) 3)
    (SENSE= (POSITION-OF-SENSED handle 3)
            (POSITION-OF-MODELLED
               right-end-effector 3))
    (FEEL right-end-effector 3)
    (SENSE= (FORCE-OF-SENSED right-end-effector 3) 0)
    (CONFIGURATION right-end-effector 3)
    (SENSE= (WIDTH-OF-SENSED right-end-effector 3)
            open)
    (GRASP ... 3 4)
    (CONFIGURATION right-end-effector 4)
    (SENSE= (WIDTH-OF-SENSED right-end-effector 4)
            (GRASP-WIDTH-FOR handle 4))
    (FEEL right-end-effector 4)
    (SENSE= (FORCE-OF-SENSED right-end-effector 4)
            (COMPLIANT-GRASP-FORCE-FOR handle 4))
```

Figure 6: Example plan output from GRIPE

GRIPE's strategy for verifying the successful execution of these two actions is: Use the position encoders of the arm to verify that the end-effector is in the correct position before and after the MOVE-TO. Before the GRASP, use the vision system to verify that the handle is in the expected location, use the force sensor to verify that the end-effector is not holding anything, and use the configuration encoder of the end-effector to verify that it is open. After the GRASP, read the force sensor and configuration encoder of the end-effector and verify that the values on these sensors are appropriate for gripping the handle. The modified plan is shown in Figure #6.

4. ISSUES

A number of issues have been raised during our development of the GRIPE system, some of which were handled in the initial implementation by making certain assumptions. In this section we examine these issues in detail and propose some preliminary solutions.

4.1. PERCEPTION VERSUS INFERENCE

The essence of verification is gathering a perception which implies that an assertion holds. A motivating assumption of our work is that inferences which have a basis in perception are more reliable as verifications than inferences (such as the specification of postconditions in an action) which are not so based. Thus our basic strategy of verification is to substitute relevant perceptions for the assertions that appear in plans.

Verification operators embody essentially one-step inferences between perceptions and assertions. There is no reason why such inferences could not be more indirect. An example appears implicitly in the GRASP action in our example above.

One of the preconditions for the GRASP action is that there must be no forces at the end-effector. Implicit in this assertion is the inference that the gripper is not holding any object when the forces at the end-effector are zero. This reasoning can be made explicit by making the assertion that the gripper is empty appear as the precondition in the GRASP action. An additional inference rule relating no forces at the gripper to the gripper being empty allows the same perception request involving the force sensor to be generated. However, now it is possible to define other strategies (e.g. using the vision system) to verify this restatement of the precondition for the GRASP action.

We intend to develop our verification knowledge base so that GRIPE can construct more complicated chains of inferences to determine how to verify the assertions appearing in plans. Under this extended verification knowledge base, there should often be several ways to verify a particular assertion. The considerations involved in choosing a verification strategy are discussed in the remainder of this section.

4.2. WHEN SHOULD THE EFFECTS OF PLAN ACTIONS BE MONITORED?

As others have pointed out [VanBaalen, 1984] [Gini et al., 1985], it is too expensive to check all the assertions in a plan. In many domains, it may be impossible. Sensors should be viewed as a resource of the agent which must be planned and scheduled just like other resources [Miller, 1985]. However, the process is aided by the observation that exhaustive monitoring may not be necessary and that selection criteria exist which can effectively limit the scope of monitoring. Some of these criteria are listed below. How they may best be combined in an assertion selection process is an open research topic.

- **Uncertainty Criteria.** Uncertainty in a number of forms may exist which requires that actions be closely monitored. This area has been the most extensively investigated [Brooks, 1982], [Donald, 1986], [Erdmann, 1985] and [Gini et al. 1985]. Uncertainty may exist in the world model which is used for planning. Uncertainty may exist about the effects of actions themselves; multiple outcomes may be possible. The effects of actions may be "fragile" and easily become undone (e.g., balancing operations). Actions may have known failure modes which should be explicitly checked. If the effects of actions have a duration, there may be uncertainty about their persistence.

- **Dependency Criteria.** There is a class of assertions which do not need to be verified at all. These are assertions which appear as postconditions of an action but are not required as preconditions of later actions, i.e., side effects. The assertions not on the this ***critical path*** of explicit dependencies between actions in a plan can be ignored in the verification process. The dependency information in the plan can be used to prune out these irrelevant effects of actions.

- **Importance Criteria.** If we have explicit representation of the dependencies among effects and actions in a plan, we can prioritize assertions for monitoring based on their criticality. The simplest metric is the number of subsequent actions which depend directly or indirectly on an assertion. More complicated metrics might take into account the importance of the dependent actions as well. The failure to achieve highly critical effects could have profound implications for subsequent error recovery.

- **Recovery Ease Criteria.** These criteria interact with the importance criteria. If an effect may be trivially re-achieved after a failure, the effects of a failure to verify even highly critical assertions is somewhat mitigated. Consequently, the need to monitor the assertion closely is not so severe.

4.3. WHICH PERCEPTION(S) CAN BEST VERIFY AN ASSERTION?

The current set of verification operators for GRIPE provide only a single, context-independent perception request for verifying individual assertions. In previous sections, we discussed how a more extensive verification knowledge base could support reasoning about multiple ways to verify assertions. These options often will be necessary.

For example, consider the difference between an arm movement which sets up a GRASP and a movement after a RELEASE. The location of the end-effector is critical to the success of a GRASP. In this case a battery of sensors such as position encoders, proximity sensors, force sensors, and vision might be indicated to verify that the end-effector is properly in place. On the other hand, a movement of an arm after a RELEASE may be performed relatively sloppily, particularly if this movement terminates a task sequence. A simple check on a position encoder (or even no check at all) may be sufficient.

4.4. HOW ACCURATELY SHOULD ASSERTIONS BE VERIFIED?

Using the same example, the latitude in the position of the end-effector for a GRASP is small; this position must be verified with a great deal of precision. On the other hand, the latitude in the position of the arm after movement away from a RELEASE is presumably quite large.

4.5. SHOULD AN ASSERTION BE VERIFIED INSTANTANEOUSLY OR CONTINUOUSLY?

In the current version of GRIPE, we assume that the successful execution of actions can be verified by instantaneously verifying the actions's preconditions before its execution, and instantaneously verifying its postconditions after its execution. This approach proves inadequate for some actions.

For example, consider a MOVE-OBJECT action which transports an object gripped by the end-effector of an arm by moving the arm. The force sensors in the end-effector should be checked continuously because the object might be dropped at any point along the trajectory.

Instantaneous monitoring is also insufficient for those actions which involve looping, for example, when a running hose is being used to fill a bucket with water. In this case, monitoring must not only be continuous but conditionalize the performance of the filling action itself.

4.6. SHOULD ASSERTIONS WITH PERSISTENCE BE RE-VERIFIED?

GRIPE's strategy for verifying assertions which are established as postconditions in one action and required as preconditions in a later, non-consecutive action is to verify the assertion twice -- both at the time of its establishment and at the time of its use.

Like the issue above, this issue concerns assertions across actions rather than assertions during actions. An error interpretation and recovery system should know as soon as possible if a condition which is needed later during the execution of a plan becomes unsatisfied. For example, suppose a part is to be heated and used in a delayed, subsequent action. If there is uncertainty about how quickly the part will cool (i.e., the duration or persistance of the "heated" assertion), then the temperature of the part should be frequently checked to verify it stays within the desired parameters. If it cools too quickly, additional heat may need to be applied before the subsequent action can be executed.

5. THE VERIFICATION LANGUAGE

The verification operators described earlier capture a restricted style of verification. In this section, we develop an extended language for verification which makes explicit a set of issues relevant to determining how to verify assertions in plans (the language is not yet implemented in GRIPE).

The need to perform both **instantaneous** and **continuous** monitoring functions suggests two fundamental types of perception requests, called **Brief** and **Prolonged**. Any particular perception request is exclusively one of these types. Brief-type perception requests handle instantaneous monitoring tasks which involve simple pattern matches against sensor or data-base values. Prolonged-type perception requests handle continuous or repetitive monitoring tasks which may involve extended modification of the plan. However, both types of perception requests may require preconditions to the use of sensors to be established by the planner. In addition, the planner ensures that any sensor resources specified are appropriate and available at the desired time. If sensor resources are not explicitly specified, the planner must choose appropriately from those available. The current GRIPE implementation does yet interact with a planner and therefore its sensor resource managment is not this facile.

Since monitoring an assertion may itself involve planning and the generation of additional plan actions, the process may recursively involve monitoring of the plan generated to achieve the original monitoring request. In the current GRIPE implementation, we allow a maximum recursive depth of two. However, to be satisfactory, we need to first, relax the depth restriction, and second, to use heuristics to constrain the recursion depth. The second requires a priori assumptions about the success of some plan operations.

```
<Perception-Request>      == <Brief-Type> |
                             <Prolonged-Type>
<Brief-Type>              == IF <Quick-Condition>
                             THEN <Action>
<Quick-Condition>         == data-base-query |
                             NOT data-base-query |
                             <Cond-Operator>
                                 <Sensor>
                                 <Value-Spec>
<Cond-Operator>           == IN-RANGE |
                             <Relational-Op>
<Relational-Op>           == < | = | > | <= | =>
<Sensor>                  == any available
                             and appropriate sensor
<Value-Spec>              == [<integer> ... <integer>] |
                             <integer>
<Prolonged-Type>          == CHECK <assertion>
                                <Time-Spec>
                                <Stopping-Cond>
<Time-Spec>               == <Time-Relationship> |
                             <Time-Designation>
<Time-Relationship>       == <Timing> <Action-Spec>
<Timing>                  == BEFORE | AFTER | DURING
<Action-Spec>             == an instance
                                of a plan-action-node
<Time-Designation>        == FOR <Time-Designation-Spec>
                             WITH FREQUENCY <integer>
<Time-Designation-Spec>   == TIME <Relational-Op> <integer> |
                             <integer> NUMBER-OF-TIMES |
                             NEXT <integer> ACTION-NODES
<Stopping-Spec>           == STOP MONITORING <Brief-Type>
```

Figure 7: Verification Planning Language

Figure #7 gives a grammar for a **Verification Planning Language** which addresses these considerations. Requests of the syntax defined in Figure #7 are generated by an expectation generator module such as GRIPE and recursively input to the planner. Eventually, this iteration flattens Prolonged-type perception requests. The final executable perception request in the plan is always of the Brief-type. For example, if a Prolonged-type perception request stated that an assertion should be monitored 5 times then the final plan would state **IF *predicate* THEN *action 5 times.*** Miller [Miller, 1985] has discussed similar ideas.

6. DETERMINING CONTEXT AND THE INTENTS OF ACTIONS

Our overall approach to verifying the execution of plans is driven by the following observation: The appropriate means of verifying the execution of an action is constrained by the intent of the action. In general, the intent of an action may vary according to context. Our results so far are restricted by the assumption that actions have a single intention. In this section, we describe our approaches to determining the intent of actions from context. They are similar to those described in [Gini et al. 1985].

One approach is top-down and assumes the existence of a hierarchical planner, as in [Sacerdoti, 1974]. Recall the example concerning two movements of an arm, one to set up a GRASP operation, and one after a RELEASE operation. The movement in the context of the

GRASP operation needs to be verified quite accurately; the movement in the context of the RELEASE operation requires only cursory verification. For example, the expanded movement operator before the GRASP might be a MOVE-GUARDED which indicates the need for careful verification; the expanded movement operator after the RELEASE might be a MOVE-FREE which requires less exacting verification.

Even when actions are not distinguished during expansion, the context provided by higher-level actions of which they are a part may be sufficient to distinguish them for the purpose of verification. In our example, the GRASP might have been expanded from a GET-OBJECT task while the RELEASE might have been expanded from a LEAVE-OBJECT task. The knowledge that the MOVE within a GET-OBJECT task is critical while the MOVE within a LEAVE-OBJECT task is not can be placed in the verification knowledge base.

The context of an action also may be determinable through a more local, bottom-up strategy. In this same example, the two MOVE actions at the lower level might be distinguished by noting that one occurs before a GRASP and the other occurs after a RELEASE. These contexts then can be used in the same way to retrieve appropriate verification strategies from the verification knowledge base.

7. INTERFACING WITH GEOMETRIC AND PHYSICAL LEVEL REASONING SYSTEMS

GRIPE reasons at what is commonly referred to as task level. We envision GRIPE and the other knowledge-based modules of our proposed PEER system interfacing with systems that can reason directly about the geometry and physics of task situations. Examples of systems are described in [Erdmann, 1985] and [Donald, 1986].

Erdmann has refined a method for computing the accuracy required in the execution of motions to guarantee that constraints propagated backward from goals are satisfiable. His approach could be incorporated into our system to generate expectations for verifying the execution of motions (only). An expectation would be a volume; a perception which indicates that a motion has reached any point within the volume would verify the successful execution of the motion.

Donald has developed a complementary technique for planning motions in the presence of uncertainities in the world model (as opposed to uncertainities in the execution of motions). He also proposes a theoretical framework for constructing strategies for detecting and recovering from errors in the execution of motion planning tasks.

8. CONCLUSIONS

The problem addressed in this paper is that of verifying the execution of plans. We have implemented a system which analyzes a plan and generates appropriate perception requests and expectations about those perceptions which, when confirmed, imply successful execution of the actions in the plan.

Typically, not all the assertions in a plan can or should be verified. We have proposed a number of heuristic criteria which are relevant to the selection of assertions for verification.

In general, verification strategies must be context-dependent; this need can be supported by an ability to analyze plans at multiple levels of abstraction.

Finally, we are developing a language for verification which makes explicit the relevant considerations for determining verification strategies: the appropriate perceptions, the degree of accuracy needed, discrete vs. continuous verification, and the need for re-verification.

9. ACKNOWLEDGEMENTS

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. We would like to thank Leonard Friedman of the USC Information Sciences Institute for instigating the PEER project at JPL and stimulating our research. Rajkumar Doshi would like to thank his advisor Professor Maria Gini, University of Minnesota, for providing inspiration and ideas.

REFERENCES

[1] Atkinson D., James M., Porta H., Doyle R.
Autonomous Task Level Control of a Robot.
In Proceedings of Robotics and Expert Systems, 2nd Workshop. Instrument Society of America
June, 1986.

[2] Brooks, Rodney.
Symbolic Error Analysis and Robot Planning.
A.I.Memo 685, Massachusetts Institute of Technology, September, 1982.

[3] Chien, R.T., Weismann, S.
Planning and Execution in Incompletely Specified Environments.
In International Joint Conference on Artificial Intelligence. 1975.

[4] Donald, Bruce.
Robot Motion Planning with Uncertainty in the Geometric Models of the Robot and Environment: A Formal Framework for Error Detection and Recovery.
In IEEE International Conference on Robotics & Automation
San Francisco, CA, 1986.

[5] Erdmann, Michael.
Using Backprojections for Fine Motion Planning with Uncertainty.
In IEEE International Conference on Robotics & Automation
St. Loius, MO, 1985.

[6] Fikes, R.E., Nilsson, N.J.
STRIPS: A new approach to the application of Theorem Proving to Problem Solving.
Artficial Intelligence Journal 2(3-4), 1971.

[7] Fikes, R.E., Hart, P.E., Nilsson, N.J.
New Directions in Robot Problem Solving.
Machine Intelligence 7, 1972.

[8] Fox, Mark S., Smith, Stephen.
The Role of Intelligent Reactive Processing in Production Management.
In CAM-I, 13th Annual Meeting & Technical Conference
November, 1984.

[9] Friedman, Leonard.
Diagnosis Combining Empirical and Design Knowledge.
Technical Report JPL-D-1328, Jet Propulsion Laboratory, December, 1983.

[10] Gini, Maria., Doshi, Rajkumar S., Garber, Sharon., Gluch, Marc., Smith, Richard., Zualkernain, Imran.
Symbolic Reasoning as a basis for Automatic Error Recovery in Robots.
Technical Report 85-24, University of Minnesota
July 1985.

[11] James, Mark.
The Blackboard Message System.
Technical Memorandum
Jet Propulsion Laboratory, 1985.
Write to author, stating reason.

[12] Miller, David P.
Planning by Search through Simulations.
PhD thesis, Yale University, October, 1985.

[13] Munson, John.
Robot Planning, Execution and Monitoring in an Uncertain Environment.
In International Joint Conference on Artificial Intelligence. 1972.

[14] Porta Harry.
Dynamic Replanning.
In Proceedings of Robotics and Expert Systems, 2nd Workshop. Instrument Society of America, June, 1986.

[15] Sacerdoti, Earl.
Planning in a Hierarchy of Abstraction Spaces.
Artficial Intelligence Journal 5(2), 1974.

[16] Sacerdoti, Earl.
A Structure for Plans and Behaviour.
Elsevier North-Holland Inc., 1977.

[17] Tate, Austin.
Planning and Condition Monitoring in a FMS
University of Edingburgh,
Artificial Intelligence Applications Institute,
AIAI TR #2, July 1984.

[18] Van Baalen, Jefffrey.
Exception Handling in a Robot Planning System.
IEEE Workshop on Principles of Knowledge-Based Systems, Denver, CO, December, 1984.
Not Published due to late submission.

[19] Wilkins, David.
Domain Independent Planning: Representation & Plan Generation
SRI, Technical Note #266, August 1982.

[20] Wilkins, David.
Recovering From Execution Errors in SIPE
SRI Technical Note #346, January 1985.