# SCENARIOS IN THE ERRAND RUNNING DOMAIN

RAJENDRA S. WALL

COINS Technical Report 81-40
December 1981

Computer and Information Science Department
University of Massachusetts
Amherst, Massachusetts, 01003

This work supported by NSF Grant #IST-8017343

## Abstract

This project report summarizes work done to investigate the theory and use of scenarios in the domain of errand running. A scenario is a compact structure used to outline the important features of the solution to some complex problem. These ideas have been used to implement a Tactical Assistant for errand running in a local community using scenarios. We discuss the implementation and use of the Tactical Assistant and the implications of the results of this work for the practicality of scenario use in other domains of interest.

## 1.0 Introduction To The Errand Running Domain

When working in the errand running domain one is given a set of goals, a physical space to perform in and a set of events that can occur randomly to disrupt the execution of actions. The task is to lay out a course of action that will accomplish the goals with a minimum of effort.

The goals in the errand running domain consist of a specific task description, such as "get groceries" or "cash paycheck," some idea of when the task should be accomplished, and how valuable this goal is. The task description indicates other information such as where to go to accomplish the goal (a grocery store, a bank) and the object of the goal (food, money).

The time specification of the goal can be vague (e. g., "sometime this morning") or exact ("appointment at 2:30"). The value of the goal gives an idea of how much consideration should go into scheduling actions to meet the goal and into worrying about events that could disrupt such actions.

The actions used to accomplish goals are for the most extent directly related to those goals. An action is scheduled to move to a location where the goal could be accomplished (e. g., a shopping center with a supermarket) followed by the action to perform the task (e. g., buy the groceries).

The physical space where the actions are performed is the local community with specific resource characteristics and topography: Where the shopping centers, etc, are in relation to "home" and each other, what shops does each contain, traffic conditions for various area, individual store peculiarities, etc. The random events or elements are those unsuspected things that can occur when trying to accomplish the goals. They are such things as the store being sold out of the item you want, the item costing more than you expected or your running out of money on a long shopping trip, or being delayed by heavy traffic on the way to an important appointment. These events may cause rescheduling of tasks, extra trips or unwanted delays.

## 1.1 Related Work

Work has been done in the field of computer planning that relates to some of the work in this paper. Cognitive studies of human errand planning led Hayes-Roth et al to propose a knowledge-source/blackboard type approach to planning [HAYE79]. Chien and Weissman examined uncertainty and Siklossy and Dreussi randomness [CHIE75], [SIKL74]. An excellent study of statistical and utility analysis methods in planning is found in [SPRO77]. McDermott has proposed a system that handles random events with no real long range planning at all, just constant rescheduling of disrupted tasks [McDE78].

In addition, work in expert systems [FEIG77] is similar in purpose (aiding humans in some complex task) to our Tactical Assistant. The knowledge in such systems is usually in the form of a large number of production rules. We will return to the use of productions rules in expert systems in a later section. In other systems, knowledge is represented as schema's or frames [MINS75]. This representation is used as the data structure for scenarios.

Military planners and analysts use abstract representations of events as the basis of their work [USAR69], [ESPO59] as do people who play conflict simulation games [PRAT81], [WANG80]. A study by Hayes-Roth examines ways in which humans generate and use such representations [HAYE80].

## 1.2 Purpose Of This Work

This work investigates several points related to the theory and use of scenarios. The first is to examine the practicality of the scenario concept. How would it work in the domain of errand running? Can we say meaningful things about future events by using scenarios? Can we use such scenarios as a basis from which to make planning decisions?

The second point is to examine the use of scenarios as the knowledge representation scheme for a Tactical Assistant - an expert system application.

The third area of investigation is the use of simulation methods to build scenarios. We need something less rigid than tree search in more complex domains, such as conflict simulation games, and this work gives a chance to test some of our ideas on generating abstract plans in a more restricted domain.

Finally we needed to examine in practice our ideas on handling random processes symbolically. Do the resulting scenarios say anything useful? Can we reflect the level of human decisions about unexpected events?

The majority of these questions we can answer positively. A complete summary is in section five.

## 2.0 Theory Of Scenarios

This section covers the ideas behind the implementation - why do we think we need scenarios, what is a scenario, where do they come from, etc.

## 2.1 Purpose Of Scenarios

In many complex domains full scale detailed solutions to problems are not possible or practical. This could be due to the inherent complexity of the domain as in oil well analysis or medical diagnosis, elements of uncertainty and chance as in errand running or weather prediction, actions of agents uncontrolled by or unfriendly to the protagonist process as in chess or GO, or a combination of all of these factors as in conflict simulation games or raising children.

In such domains a solution is usually some kind of a plan and a way to implement the plan: A detailed series of actions to deal with or solve the problem. In less complex domains, such as the AI "blocks world," one is easily able to produce a detailed plan of action: Every single action, its consequences and side effects can be plotted out and taken care of; a list of actions produced that some effector process of menial intelligence can then carry out. At worst there may be some conditional actions or multiple choice nexus.

Attempts to derive similar levels of solutions for the complex domains listed above have evoked frustrated cries for more processing power or better mathematical models. Rather than attempt to produce such detailed solutions we propose an alternative of creating a <u>scenario</u>: a form of abstract plan representation outlining the consequences of a course of action.

This scenario can then be used to decide courses of action, outline contingency plans or force reexamination of goals set in the problem statement.

### 2.1.1 What Is A Scenario?

A scenario is collection of projections of future events, each of which is based on a set of assumptions about the behaviors, intentions and effects of the various processes involved. Each projection depicts the course of events in this interpretation of the future - the important events or actions taken by the various processes involved.

Stated another way, the total set of possible future world states is in effect divided up into classes or categories on the basis of these assumptions about the processes involved. From each class one or more examples are chosen to illustrate the kinds of events that could occur in that class of worlds.

Scenarios are based on goals. In each domain where scenarios are applicable there is some goal or set of goals to be secured. These goals are statements about the state of the world under consideration. They can represent situations to be instantiated or maintained. A set of goals could contain interrelated or even contradictory statements.

Each goal has a value assigned by some process or user that generated the goal that reflects the worth of this goal to the assigning process. For example, in the errand running domain, the value of one goal of buying a newspaper might be low while another of attending a dentist appointment when you have a toothache might be very high.

Actions that effect high value goals deserve more attention than do those effecting low value goals when considering the consequences of courses of action. Random events of high probability deserve more attention than those of low probability. Sensible actions taken by unfriendly processes to disrupt friendly plans deserve more attention than irrational actions.

This then is the basis of scenarios: In response to a set of goals a set of courses of action to achieve the goals is proposed. Consideration is then given to the effects of random or unfriendly processes. This consideration can be light or intense depending on the worth of the goals or pessimism of the system, controlled by a threshold to be set at various levels of concern, each setting prompting a review of the actions. Wherever deemed important the effects of random or unfriendly processes are noted and the course of action revised.

A body of information is thus produced which is called a scenario. It consists of a set of sub-scenarios each built with a different setting for the concern about random or unfriendly process threshold. It shows in a clear concise way the results at some future time the effects of actions taken in the present.

### 2.1.2 Where Do Scenarios Come From?

A scenario is basically a set of statements about the road to the solution of the problem. In the errand running domain, as in other planning tasks, this means a set of statements about the future. There are three basic ways to create these sets of statements: static analysis, simulation and retrieval of experience.

Static analysis means looking at the current situation and having enough knowledge about the domain to immediatly draw some conclusions about future events. This is what most AI planning systems of the past have done [NILS80]. The world they deal with is simple enough that there is an exact and constant goal/action linkage - a given goal can always be effected by some specific action. There are also no random or unfriendly processes. Thus such systems can postulate series of actions as easily as stacking up blocks.

Static analysis can also be used to a limited extent in more complex domains, such as predicting outcomes of battles [DUPU79] given a staggering amount of data, or by limiting the specificity of the statements produced. We do not examine the generation of scenarios by static analysis in great detail in this work or later reports, prefering to concentrate on the other two, "more interesting" methods.

Simulation involves examining the current situation and using knowledge about the ways in which processes function in the domain to project the results of operation of those processes on the current situation. This requires a model of the world - what things can change and how, as well as a model of each active process - what it can change, when, how, and in the case of unfriendly processes, why. This method is used by most game playing systems to decide which move to make [SLAT77]. Another good example of simulation is Wesson's Air Traffic Control system [WESS77]. Of course, neither of these produces exactly what we have called a scenario, but the method of using knowledge about the fuctional qualities of the domain to predict the future behavior of the system is similar. The main difference is that these systems are trying to generate an exact, detailed, minute-by-minute, move-by-move map of the future while the idea behind the scenario is to produce a summary of the critical or noteworthy events.

One of the purposes of this work was to examine the problems of using simulation to generate scenarios in the errand running domain.

The last method, and perhaps the most interesting, is retrieval from experience. This involves having a large body of experience in the domain: When a new situation is encountered the data base of experience is examined and a situation similar to the current one is recalled. Along with the remembered situation is knowledge about what happened when certain things were tried - what worked, what didn't, etc. The experience is then an example used to illustrate the current situation and possible futures. A more complex examination of this method will be performed in later reports.

## 2.2 The Tactical Assistant

One of the uses of scenarios examined by this work is as a representaional aid for a Tactical Assistant. This section explains the concept of the Tactical Assistant in the context of the domain of interest, errand running.

### 2.2.1 What Is A Tactical Assistant?

A Tactical Assistant is a form of expert system designed to help a user, who has high level strategic goals to accomplish but is unfamiliar or unconcerned with the low level detail of a domain, decide a specific course of action.

The user proposes a set of goals. The Tactical Assistant takes this set and generates a scenario describing the possible outcomes of actions taken to achieve the goals. The unsatisfied user may change the goal set or perhaps make suggestions about possible courses of action. The Tactical Assistant then generates a revised scenario. This iterative process continues until the user is satisfied with the outline of the future presented, or gives up.

In the errand running domain this process can be thought of as the user first making a list "these are the things I would like to do today". The Tactical Assistant, with its knowledge of the town, traffic patterns, etc, generates a scenario laying out the most convenient way to accomplish the goals, noting any conflicts or problems that may arise. This scenario is reviewed by the user, who then may reschedule, postpone or cancel proposed actions, in turn causing the Tactical Assistant to revamp the scenario. In a sense the user is asking a set of "What if?" questions which the system tries to answer.

## 2.2.2 How Does The Tactical Assistant Work?

The Tactical Assistant as an expert system has the detailed knowledge about the domain that must be used in making predictions. This knowledge includes the tactics available in the domain, a detailed model of the domain, and detailed models of random or unfriendly processes operating in the domain.

Each tactic describes a set actions designed to meet a particular goal. Any goal may have numerous applicable tactics. Any particular tactic may affect more than the designated goal favorably or adversely. The Tactical Assistant must understand and be able to handle such side effects.

Once a tactic or a set of tactics to achieve each goal has been chosen the entire set of tactics to achieve all the goals and the order to apply them is called a <u>course of action</u>. For any given set of goals there may be different courses of action available.

The Tactical Assistant must examine the available courses of action, weigh the costs and benefits, note any possible problems with random or unfriendly processes, and summarize the courses of action as a scenario for the user.

The Tactical Assistant would also aid the user during execution of the suggested plan. If something "went wrong" it would still have all the alternative courses of action available to recover and continue towards the goal.

## .3 Desiderata For The Tactical Assistant

Goals for our implementation of a Tactical Assistant for the errand running domain were the following:

- o Take an arbitrary list of goals consisting of things to be accomplished.
- o Accept values rating the importance of each goal.
- o Accept specific times of when the tasks are to be performed.
- o Model three different random processes: heavy traffic, desired item sold out and insufficient funds for purchase.
- o Include conditional probabilities in random process models as functions of space and time.
- o Allow activation of concern about random processes to be determined by threshold.
- o Generate courses of action to accomplish the goals.
- o Implement the errand running tactics: meet-appointments, highest-value and minimize-travel.
- o Generate courses of action by using the tactics in a hierarchy: meet-appointments, highest-value, and minimize-travel.
- o Summarize courses of events as possible world projections.
- o Use scenarios to highlight possible goal conflicts or random effects.
- o Show most convenient course of action first.
- o Track usage of pertinent resources (e. g., money).
- o Track space and time relationships among goals to allow easier comparison of courses of action.
- o Save the alternative courses of action for use as needed during execution of the suggested course of action.

## 3.0 Implementation

Implementation of the Errand Running Tactical Assistant was done in the version of LISP available on the UMASS Cyber 175. All of the above goals were met, within the memory limitations of the host computer.

## 3.1 The Working System

The working system accomplished all of the desiderata. A user was able to define a set of goals with values and desired times of completion. The system used simulation to generate scenarios from courses of action based on tactics incorporating basic errand running heuristics. Three random processes were simulated with conditional probabilities and thresholds: heavy traffic, desired item sold out and insufficient funds. A small front end was built to facilitate user communication.

### 3.1.1 How The System Works

The implemented system is a four phase process. First is goal definition and tactic initialization. Second is action conflict recognition and resolution. Third is the actual scenario organization. And fourth is the presentation to the user.

3.1.1.1 Phase 1 - Goal Definition And Tactic Initialization - During the first phase of processing the user is asked to define the goals of this run.

- The object of the goal must be specified e. g., "food" for the goal "get groceries".

- The value of the goal - a worth rating from one to ten.

- The desired time of completion of the goal.

The time of completion can be specified as either a suggested time or a mandatory time. Specification of a mandatory completion time forces the system to accomodate that goal at exactly that time, while a goal with a suggested time will be fit in wherever convenient as close as possible to the suggested time.

The system then examines the tactics available to accomplish each goal. These are then instantiated in a graph structure using the graph processing language called Grasper [LOWR79]. Linkages are then made among the actions of the tactics noting the following relationships: Spatial, temporal, value and conceptual.

Spatial links are made between any actions that could take place in the same sector [see 3.1.2.1]. This will aid later phases that use errand running convenience heuristics in clustering actions. Links are also made between actions that could take place in adjacent sectors. Such links are used to order actions to minimize travel time.

Temporal links are made to express six different relationships between two actions on the basis of their desired times of completion. The first three relationships consider time specifications that are exactly the same. The first is between two actions that have the same mandatory times of completion. This can lead to an irresolvable conflict that must be handled by special means during phase 2.

The second is between a mandatory time and a suggested time. This link alerts later processes that the suggested time action will have to be done earlier or later. The third type of temporal link is between two suggested time actions. One or the other (or perhaps both, depending on other links) will have to be performed at another time.

The last three temporal links are similar, but between actions that have time specifications "near" each other (within an hour as presently implemented). So again we have mandatory-mandatory, mandatory-suggested and suggested-suggested links. These links allow the system to check for indirect conflicts that may be caused by travel time from one sector to another.

Value links are made to allow the system to try to accomplish as many high value actions as it can first. The links represent a partial ordering among the actions based on value, in effect sorting the actions.

Conceptual links are made between nodes that are related by some implementation designated concept. The implemented concepts that were linked were such things as use of money, type of action (movement, purchase) and object of goal. For example, an action of buying a newspaper would be linked to an action of going to the bank, since each involved a monetary transaction, even though one used up money and the other replaced it.

This complete graph structure is then saved for use by the second phase.

3.1.1.2 Phase 2 - Conflict Recognition And Resolution - During the second phase of processing the system uses the linkage network set up in the first phase to sort out incompatible actions. This was done on the basis of the limitation that you can not be in two places at once. Such a requirement was noted by the class of temporal link between actions with mandatory accomplishment times that were the same or too near each other.

Processing proceeded by first examining all the actions to complete Mandatory Accomplishment Time Goals (MATG's) in order of decreasing value. A subspace was created and the first such action placed in it, along with all other potentially compatible actions. Only actions with direct conflicts with this action were left behind. Note that the set of potentially compatible actions was compatible with the initial high valued action under consideration and not neccessarily internally compatible.

If there were any actions left, a second subspace was created and the next highest value mandatory accomplishment time action was placed in it, along with all of its potentially compatible actions. This process of creating subspaces and filling them with actions continued until the original set of action had been completely examined. If there had been no MATG actions in the beginning then all the actions would have been placed in the first subspace.

Now each subspace was examined for internal conflicts. If any MATG action conflicts existed, the subspace was split into two new subspaces each with a copy of all the non-conflicting actions and one of the actions in conflict. These new subspaces were then placed on the list of subspaces to be checked for internal conflicts.

Eventually a set of subspaces was produced that had no internal conflicts. Each subspace contained a set of actions that should be compatible with each other. The next step was to combine them into a course of action.

3.1.1.3 Phase 3 - Course Of Action Organization - Tactics have been outlined to accomplish each goal. Incompatible actions from those tactics have been identified and the conflicts resolved. At this point the actions have not been finalized. This means, for example, that an action to go buy groceries has been selected but which grocery store to patronize has not.

This phase of processing examines each subspace produced by phase 2 and constructs an explicit course of action from it by instantiating course-of-action links among the actions. Due to memory limitations only the course of action representing the use of the chosen plan heuristics was actually linked while the other altermative courses were only outlined. Recovering explicit detail of alternative courses requires running a set of routines to change the planning assumptions and re-running this phase.

The examination of each subspace begins by looking for actions to achieve mandatory accomplishment time goals (MATG's). The highest valued such action is chosen to be one of the first anchor points. The other initial anchor point is the action "first" which is merely to be at home to begin a day of running errands.

The high valued action is checked to see which of its choice of location instantiations has the highest valued action cluster. An action cluster is made up of all the actions of non-mandatory completion time that could be accomplished in that sector. The value of an action cluster is the sum of the values of the actions in that cluster. Thus each location in which the high valued action could be performed is checked to see what other actions could be performed in that same sector "at the same time." Those other actions are clustered and their values added up. The location with the highest valued cluster is the one chosen for performing the high valued action. If two or more clusters have the same value then the location that minimizes travel time is chosen.

The complete action cluster with the high valued action along with the other half of the errand running tactic, movement to the designated location, is then instantiated in the course of action linkage. At this point the effects of random processes are examined.

Each random process model consists of a set of absolute and conditional functions that examine the course of action and check for applicability. Certain processes only affect certain actions, e. g., heavy traffic only affects movement. The process model functions produce a percentage probability of the event taking place, given the locale, the store, the time of day, etc which is then checked against the threshold of concern for this sub-scenario. If the threshold is exceeded then the appropriate action is taken to note the effects of this event on the planned course of action. This threshold of concern can be set to any level by the user of the Tactical Assistant.

After all the random processes have been applied processing continues with the examination of the actions that have not yet been included in the course of action. Again the highest valued action is taken and run through the above described processing.

Eventually all the actions are linked into the course of action for this subspace. This subspace, its course of action, all the events posted by random processes and the level of concern used to select random events are now called a sub-scenario.

The processing of this phase continues by examining the next subspace and performing the same construction and analysis on it as was described above. When all the sub-scenarios have been produced the entire set of subspaces, linkages, notes, etc, now called the complete scenario, is then passed to the next phase for final processing and presentation to the user.

3.1.1.4 Phase 4 - Presentation Of The Scenario - The last phase of processing takes the complex structure created during the previous three phases and presents it in a simple form to the user.

The routines implemented at the present are not very clever or spectacular. There are routines to review the goals of this run, the primary courses of action developed, and the effects of random processes. The primary course of action is the one that was built using the basic planning heuristics of least travel, highest valued goal first, etc. In addition routines were implemented to save such information in disk files for collection and preservation in hard copy form.

To see the complete set of alternative courses of action requires use of Grasper print functions. The resulting display is awesome and overwhelmingly complex to the naive user but since no such users were using this version of the system it was decided to not pursue more accessible display and interaction routines at the present time. This lack of user oriented display routines would also make it difficult for the naive user to use the generated scenarios as an aid during "execution" of the plan.

## 3.1.2 Data Structures

This section describes the data structures used to represent the physical layout of the community, the goals, the actions and the random processes as well as the choices of goals, tactics and random elements that were implemented.

### 3.1.2.1 Physical Layout

- The internal map used by the system was a representation of the local Northampton area. The area was divided into regions called sectors, each representing a recognized section of the community.

Five sectors, including the home sector, were implemented. They were "shopping center one" (coded s-ss1) representing the Kingsgate Plaza, "shopping center two" (coded s-ss2) representing the Hamp Plaza, downtown Northampton (coded s-dtn), the Hadley Malls (coded s-mls) and home (coded s-home).

Choice of which shops to include in the representation of each sector was made arbitrarily in order to promote variety and test the system envelope. Choice of which tasks could be accomplished at which locations was made similarly. For example, although there is a grocery store near the Hadley Malls it was not included in the s-mls representation. For the most part however, the representation is accurate. For simplicity, each shop represented was given a single task that could be accomplished there. Figure 1 shows a map of the community.

### 3.1.2.2 Goals And The Goal Structure -

The types of goals available consisted mainly of accomplishing some transaction or purchase. Items that could be purchased included food, newspapers, shoes, tools, and toys. In addition one could request a trip to the bank to cash a paycheck or one could attend the movies.

Food could be purchased at either of the two grocery stores, one in each shopping center. Newspapers could be purchased in any of the bookstores located in all of the non-home sectors. Shoes were available in department stores located in the shopping centers and the malls. Tools could be purchased only at the hardware store in shopping center one. Toys were only available at the toy shop downtown. There were three branches of the bank: Downtown, shopping center two and the malls. Movie attendance was limited to the pictures showing at the mall cinemas.

The goal data structure was implemented as the LISP form of a frame, an association list. The slots for each goal were:

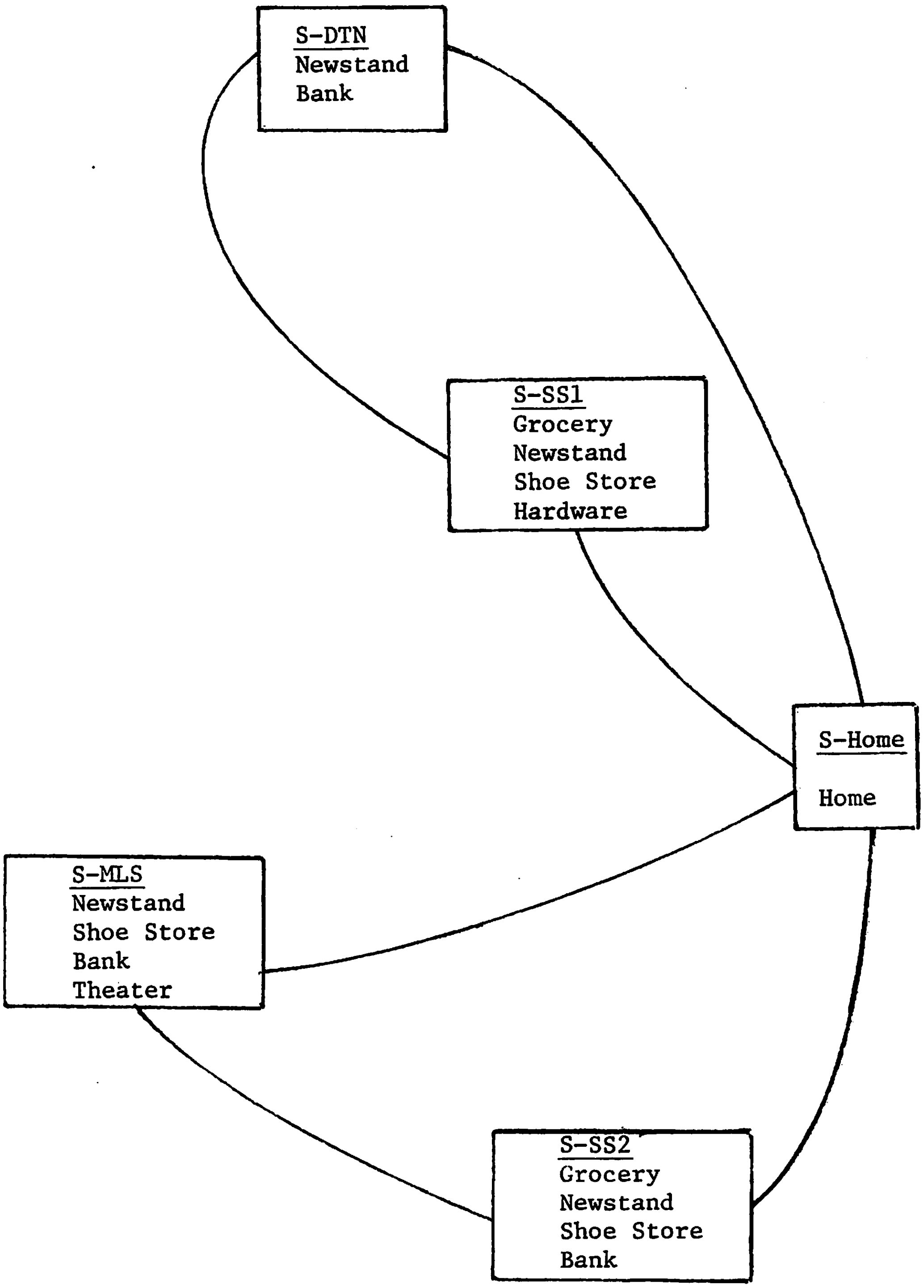

Figure 1: The Local Community

1. The name.

2. The "due date" - the time of completion and specificity (mandatory/suggested).

3. The value of the goal.

4. The location of the goal (as a class, e. g., "grocery store x").

5. The object of the goal (e. g., "food").

6. The goal verb (e. g., "get").

7. The suggested type of action (e. g., transaction).

#### 3.1.2.3 Tactics And The Action Structure -

The tactics of the errand running domain were not very complicated. Nearly all the tactical knowledge was removed from the tactic bodies and placed in a set of heuristic planning rules that would be applied to all the tactics.

These planning rules included "leave home an hour before the specified completion time of the goal," "try to perform as many high valued goals as possible," "given a choice complete the goal at the location closest to home," "try to accomplish as much as possible in each trip," and "avoid if possible heavy traffic areas." Tactics then consisted of the actions to actually perform the tasks at the correct locations plus a movement action to get the user there.

The data structure used for the actions was basically the same as that used for the goals: A frame with slots for name, "due date" of corresponding goal, type of store to perform action, item desired (goal object), the action verb and the type of action. In addition, when instantiated into the course of action linkage, the action was given a specific sector value for where it would take place plus a specific time for when it would take place.

3.1.2.4 Random Elements And Their Models - There were three random elements implemented:

1. Heavy traffic. Heavy traffic could occur while the errand runner was travelling from one place to another. If it occured then the traveller would be delayed.

2. Item sold out. Item sold out was a random event that could occur at any store other that the bank. It meant that the store no longer had in stock the item the user was looking for.

3. Insufficient funds. Insufficient funds could be thought of in two ways. One was that the price of the item turned out to be more than the user was willing to pay. The other was to think of insufficient funds as that the price of the item turned out to be more than the user was able to pay.

3.1.2.4.1 The Model - Each random element model was represented as a frame or association list similar to the structure of the goals. The slots in each random element frame were:

* The name of the element.
* The action of interest.
* The action estimation function.
* The initial estimate pairs.
* The conditional factor list.
* The function expressing the effects of the element.

The value of the name slot was simply the name of the element. The action of interest refered to the type of action that this element could affect, e. g., heavy traffic affects movement actions.

The action estimation function was the name of a function that when applied to the action would return a value to be used to determine the initial estimate and the conditional factors.

The initial estimate pairs slot was filled with a list of pairs. Each pair consisted of a name and a percentage value. The name was matched against the value produced by the action estimation function to determine the initial probability estimate. For example, for the random element item-sold-out the action estimation function was the location of the goal - what kind of a store it was. This value was then used to find a match in the initial estimate pairs to give a first rough guess of how often that store would be sold out of that product. It was at this point that it was specified that a bank would not run out of money.

The conditional factors slot was also a list of pairs. The first element of the pair was a conditional expression concerning the world state, the time of day, the errands performed so far, etc. The second element was a list of conditional change pairs similar to that of the initial estimates in that the first item was a name to match the action estimation function but the second item was a function to be evaluated. The result of this function was the conditional estimate.

The random element effects slot was filled by a function that would be evaluated to determine the results of the action of this random element on the course of action.

Use of the random element models consisted of taking each proposed action in the course of action and "applying" each random process to it. This "application" meant first that the random element's action of interest was checked against the type of the action under consideration. If there was a match it meant the element could affect this action and further concern was necessary.

Next the initial estimate was made by first applying the action estimation function to the action and then using the result to find a match in the list of initial estimate pairs.

The third step was to check for any changes in the probability estimate due to conditional factors. This was done by evaluating the first element, the conditional expression part of each conditional factor pair. If it evaluated to True then the list of conditional change pairs was checked.

The list of conditional change pairs was checked for a match of the first element of the pair to the result of the action estimation function. When a match was found the second item of the pair was evaluated. The result was the conditional factor estimate and was added to the initial estimate to give a total percentage value of the probability of the random event taking place.

This percentage probability was then checked against the given threshold of concern. If it was greater then the event was considered to have occured and its effects must be reckoned with. This was done by evaluating the random effects function of the random element. All results were applied immediately.

3.1.2.4.2 The Implemented Elements - Each implemented random element had a frame with all the slots, values and functions filled as described above.

For the random element <u>heavy traffic</u>, the action of interest was movement. The action estimation function returned the destination of the movement. This allowed an initial estimate to be made of the general chances of running into heavy traffic during the movement. Figure 2 shows the heavy traffic frame.

```
(frame
 (re-heavy-traffic
  (NAME (VALUE (re-heavy-traffic)))
  (RE-ACTION-OF-INTEREST (VALUE (a-move)))
  (RE-ACT-EST-FNC (VALUE (Destination-of-Action)))
  (RE-INIT-EST-PAIRS (VALUE (((s-ssl 30) (s-home 10) (s-ss2 40)
   (s-ss2 40) (s-dtn 70) (s-mls 60))
  (RE-COND-FACTORS-PAIRS)))
   (VALUE
    (((AND (GREATER (TIME) 30) (LESSP (TIME) 42))
     ((s-55 (20) (s-home 10) (s-ss2 20) (s-dtn 30) (s-mls 30)))
     )
    )))
  (RE-EFFECTS (VALUE (Re-FX-Delay)))))
```

Figure 2: The Heavy-Traffic Random Process Frame

The conditional factors affecting the chance of heavy traffic were based on the time of day in which the movement was occuring, for example, the chances were greater during rush hour. Certain areas still had higher conditional factor changes than others so again the destination of the action was considered to determine which conditional change pair to use.

After the initial and conditional estimates had been added up the result was checked against the threshold of concern. If the probability exceeded the threshold then the effects of heavy traffic were applied. The initial version of the Tactical Assistant posted the delay caused by the heavy traffic and checked to see if this affected any mandatory completion time goals. If so, the time of the movement action was moved up earlier if possible so as to allow completion of the tasks. The functions to handle the ramifications of these changes were too large to fit in with the rest of the system and were replaced in the final working version with a function that posted a warning to the user.

The item-sold-out element was concerned with transactions. The action estimation function checked the location of the goal - the store where the transaction was occuring. The initial estimate pairs reflected how likely it was that that store would be arbitrarily sold out. Banks were deemed to have and infinite supply of money. The conditional factors revolved around the time of day. It was assumed that each store had a full supply of the item in the morning and gradually sold them during the day. Some items, like the morning newspaper, would go fast early, while others, like movie tickets, would be sold as show time approached.

If the total probability estimate exceeded the threshold of concern then the effects of the item being sold out were considered. The initial system had functions to handle the effects by trying to schedule an alternative purchase at another location. Only one other try was allowed. These functions were also reduced to posting a warning in the final system.

The last random element implemented was insufficient funds. The action of interest was again transaction and the action estimation function was the location of the goal. Using the store as the determiner of price worked since each store only carried one item.

The initial estimates reflected assumptions about the prices of items available. The conditional factors were based on whether or not the user had been to the bank before this action. If so, then is was decided that the user had acquired an effectively unlimited supply of funds and this element was of no concern. If not then the conditional change function in effect added up the money spent on purchases made up to this point to determine the chance of running out of money for this purchase.

If the probability estimate exceeded the threshold of concern then the effects of not having enough money were considered. the initial system would try to schedule a trip to the bank into the course of action. This would, in some cases, necessitate a near complete reorganization of the plan. In the final working system this was reduced to telling the user to post such a trip as a goal and rerun the system.

## 4.0 Experiments

Experiments were run on the Tactical Assistant to see if it could perform as expected. They included experiments with two, three, four and seven goals. The experiments were run to each produce a scenario that consisted of projections for each of five different thresholds of concern.

The thresholds represented different attitudes about the future. The first was set "unrealistically" high, producing an over optimistic "perfect" view of a future where nothing goes wrong. The second was the other extreme, set "unrealistically" low it produced an over pessimistic "Murphy's Law" view of a future where everthing goes wrong. In between these two extremes were three "reasonable" settings that gave somewhat more moderate views of the future.

It was felt that a Tactical Assistant that produced a scenario consisting of these five projections could give the user a complete and well rounded view of the possibilities. The two extreme views were presented first to give and idea of the bounds of the scenario space. The next three projections presented gave a "middle of the road" view. If the user were habitually overconfident or overcautious then the extreme views could be thought of as the most reasonable.

Experiment one was the initial test of the system, its use of the planning heuristics, sychronization of events, use of thresholds and the effects of random processes, etc. It was done with two goals specified by the user. The system was able to handle this, showing the user the five different projections.

Experiment two was done with three goals. The times of completion were widely separated. One is able to see, from the five projections, the possible effects of random events.

Experiements three through five involved the three goals due at the same time. The difference in the experiments was the degree of specificity of the goal due times. Number three had all three goals with the same mandatory due time. This forced each action into a different subspace. The user then had a choice: either only one of the goals could be accomplished or the due times/degree of specificity of some of the goals must be changed.

Experiment four was a change in one of the goal time specificities to suggested. The actions are now only split into two sub-spaces, each a choice of performing one of the conflicting actions plus the non-mandatory action.

The last of this series of experiments, number 5, has only one mandatory goal time. Here all the actions were compatible and no splitting was necessary.

Experiment six was run with four goals. It had two sets of conflicting mandatory time goals, the first pair occuring early and the other later enough to avoid any time conflict.

The scenario produced shows the breaking of the main group of actions into four subchoices - one bifurcation caused by the early time conflict and the other by the later time conflict.

The last experiment, number seven, was run with all seven possible tasks requested. The system had to make travel decisions, timing decisions, etc. The results show the system in its full working glory, as well as the problems caused by strict observance of the heuristics as they were stated. To get a "more reasonable" plan would require either a different ordering of the heuristics or else a larger and more clever means of applying them.

## 4.1 An Example

As an example, Figure 4b shows the complete scenario from experiement two. The goals are to

1. Purchase groceries,

2. Buy a hammer and

3. go to the movies.

Five segments are shown depicting five different assumptions about the future. Using simulation, these assumptions become threshold values against which the random event probabilities are tested. The threshold is shown as a percentage, thus any value over 100 means no random event will occur (none can have a probability greater than 100%), any value of zero of less means anything (and everything) can happen, and values in between reflect that amount of concern about the random events.

```
;first, the goal frames supplied by the user denoting tasks
;to be accomplished are presented.

"The goals in these segments:"
(get-groc
   (NAME (VALUE (get-groc)))               ;name of the goal
   (DUE-DATE-OF-GOAL (VALUE ((d-sug 3)))) ;when to be
                                         ;accomplished: "3"
   (VALUE-OF-GOAL (VALUE  (3)))          ;worth of goal: "3"
   (LOCATION-OF-GOAL (VALUE ((l-groc x))));type of store
                                  ;necessary: "grocery"
   (TYPE-OF-GOAL-OBJECT (VALUE (i-food))) ;what item is
                                          ;needed "food"
   (TYPE-OF-GOAL-VERB (VALUE (v-get)))    ;how to acquire
                                          ;item "get"
   (TYPE-OF-GOAL (VALUE (a-transact)))    ;class of
                              ;goal: "transaction"


  (get-tool
   (NAME (VALUE (get-tool)))              ;name
   (DUE-DATE-OF-GOAL (VALUE ((d-sug 10))));time due
   (VALUE-OF-GOAL (VALUE (9)))            ;value
   (LOCATION-OF-GOAL (VALUE ((l-hdw x)))) ;where to go
   (TYPE-OF-GOAL-OBJECT (VALUE (i-tool))) ;what to get
   (TYPE-OF-GOAL-VERB (VALUE (v-get))     ;how to do it
   (TYPE-OF-GOAL (VALUE (a-transact))))   ;class


  (go-to-movie
   (NAME (VALUE (go-to-movie)))              ;name
   (DUE-DATE-OF-GOAL (VALUE ((d-mand 20)))) ;time due
   (VALUE-OF-GOAL (VALUE (8)))               ;value
   (LOCATION-OF-GOAL (VALUE ((l-movies x))));where to go
   (TYPE-OF-GOAL-OBJECT (VALUE (i-movie)))  ;what to get
   (TYPE-OF-GOAL-VERB (VALUE (v-get)))      ;how to do it
   (TYPE-OF-GOAL (VALUE (a-transact))))     ;goal class
 )


(SEGMENT segmentxaaa)
(Random Threshold 10000) ;Set high so no random
                         ;events are considered


(PROJECTION projectionxaaa)         ;The projection of the
                              ;future with this
                              ;assumption about the
                              ;effects of random
                              ;processes.
(ACTION first OCCURS AT s-home)  ;All projections begin
                                 ;at home
(TIME 1)                         ;The beginning of the day
(ACTION movexaab:goto-s-ss2 OCCURS AT s-ss2);Travel
                                 ;to shopping center 2
(TIME 7)
(ACTION actionxaaa:get-groc OCCURS AT s-ss2);purchase food
(TIME 8)                         ;purchase only takes one
                                 ;time unit
(ACTION actionxaab:get-tool OCCURS AT s-ss2);purchase tool
(TIME 17)                    ;begin trip to mall
(ACTION movexaaa:goto-s-mls OCCURS AT s-mls);travel
(TIME 20)                    ;Time movie begins
(ACTION actionxaac:go-to-movies OCCURS AT s-mls);Watch movie


(SEGMENT segmentxaab)
(Random Threshold 0) ;Lowest setting "everything goes wrong"

(PROJECTION projectionxaaa) ;Projection for the
                        ;"Murphy's Law"
                        ;random process assumption
(ACTION first OCCURS AT s-home); begin at home
(TIME 1)
(ACTION movexaab:goto-s-ss2 OCCURS AT s-ss2)
("action:" (movexaab) "may be delayed due to heavy traffic")
```

```
                        ;warning of effect of random process.
(TIME 7)
(ACTION actionxaaa:get-groc OCCURS AT s-ss2)
("action:" (get-groc) ("may fail to achieve goal.  Retry
  should be made at location(s):" (s-ss1)))
                        ;This grocery may be out
                        ;of one of the items desired.
(TIME 8)
(ACTION actionxaab:get-tool OCCURS AT s-ss2)
("action:" (get-tool) "may fail to achieve goal and no retry
  is possible")          ;There are no other hardware stores.
(TIME 17)
(ACTION movexaaa:goto-s-mls OCCURS AT s-mls)
("action:" (movexaaa) "may be delayed due to heavy traffic")
(TIME 20)
(ACTION actionxaac:go-to-movies OCCURS AT s-mls)
("action:" (go-to-movie) "may fail to achieve goal and no
  retry is possible") ;This movie is not showing anywhere
                      ;else in town



(SEGMENT segmentxaac)
(Random Threshold 75) ;an intermediate setting

(PROJECTION projectionxaaa);projection of a
                        ;"reasonable" assumption
                        ;about foreign processes.
(ACTION first OCCURS AT s-home)
(TIME 1)
(ACTION  movexaad:goto-s-ss2 OCCURS AT s-ss2)
(TIME 7)
(ACTION actionxaaa:get-groc OCCURS AT s-ss2)
(TIME 8)
(ACTION actionxaab:get-tool OCCURS AT s-ss2)
(TIME 17)
(ACTION movexaaa:goto-s-mls OCCURS AT s-mls)
(TIME 20)
(ACTION actionxaac:go-to-movies OCCURS AT s-mls)



(SEGMENT segmentxaad)
(Random Threshold 50)     ;Halfway between perfect
                          ;and terrible.

(PROJECTION projectionxaaa)  ;Another projection to outline
                          ;intermediate possibilities
(ACTION first OCCURS AT s-home)
(TIME 1)
(ACTION movexaaf:goto-s-ss2 OCCURS AT s-ss2)
(TIME 7)
(ACTION actionxaaa:get-groc OCCURS AT s-ss2)
(TIME 8)
(ACTION actionxaab:get-tool OCCURS AT s-ss2)
(TIME 17)
(ACTION movexaaa:goto-s-mls OCCURS AT s-mls)
```

```
("action:" (movexaaa) "may be delayed due to heavy
  traffic");Malls will sometimes become crowded
          ;near the time the movies start.
(TIME 20)
(ACTION actionxaac:go-to-movies OCCURS AT s-mls)



(SEGMENT segmentxaae)
(Random Threshold 25)    ;Assume most things go wrong

(PROJECTION projectionxaaa)
(ACTION first OCCURS AT s-home)
(TIME 1)
(ACTION movexaah:goto-s-ss2 OCCURS AT s-ss2)
("action:" (moveaah) "may be delayed due to heavy traffic")
(TIME 7)
(ACTION actionxaaa:get-groc OCCURS AT s-ss2)
("action:" (get-groc) "may fail to achieve goal due to lack
  of money."
  "please schedule trip to the bank and re-run.")
                      ;Rather than the item being
                      ;sold out, the groceries
                      ;could cost too much.
(TIME 8)
(ACTION actionxaab:get-tool OCCURS AT s-ss2)
                          ;No trouble
                          ;buying the tool.
(TIME 17)
(ACTION movexaaa:goto-s-mls OCCURS AT s-mls)
("action:" (movexaag) "may be delayed due to heavy traffic")
(TIME 20)
(ACTION actionxaac:go-to-movie OCCURS AT s-mls)
("action:" (go-to-movie) "may fail to achieve goal and no
  retry is possible")
```

Figure 4b: Output from the simulation based Tactical Assistant

Examples of some the other experiments are found in the appendix.

## 5.0 Conclusions

This section reviews the results of the implementation and makes suggestions for further versions, discusses the worth of this work in terms of the goals set out when when it began, and finally looks forward to what should be done next.

## 5.1 Implementation Review

As was seen in the discussion of the experiments, the system performed as expected. Actions were suggested by tactics to accomplish goals, conflicts among those actions were sorted out and resolved, courses of action were decided upon, random effects considered, and the final results given to the user. It worked to meet mandatory accomplishment time goals, perform high value actions, minimize travel time, etc, occasionaly producing a strange plan due its heuristics.

Improvements might include a more detailed simulation, more errands, more products per store, more random elements, etc, but perhaps the most worth while change deserving further investigation is the ordering of the planning heuristics. Perhaps some sort of meta-planning statements about how to make decisions could be provided and manipulated. Unfortunately, such investigation is beyond the scope of this work.

## 5.2 What Did This Work Accomplish?

A number of questions were asked in the begining of this paper in the section on the purpose of this work, and is was suggested that four points needed investigation:

1. The practicality of the use of scenarios.

2. Use of scenarios as the knowledge representation scheme for a Tactical Assistant.

3. Use of simulation methods to build scenarios.

4. The practicality of treating random processes symbolically in scenarios.

### 5.2.1 The Practicality Of The Scenario Concept

It is felt by the author that building an expert system that makes planning suggestions in the form of scenarios demonstrates the practicality of using scenarios. To some extent this is a "proof by construction" in that we built a system that would create scenarios in order to show that the creation of scenarios was useful. The key is whether or not the output, the scenarios created, appear useful. We feel that within the limitations of the host machine and the resulting limit on detail the scenarios produced are a good look at the future.

The range of possibilities suggested by the set of sub-scenarios produced allows the hypothetical user to get a good idea of what could be in store and allows planning decisions and goal specifications to be reviewed and restructured without having wasted any time actually running around.

By showing the practicality of the use of scenarios as an aid in making planning decisions it is felt that a case has been made for the use of scenarios as aids in the complex domains suggested in section 2.1. Even though a domain such as oil well analysis or program synthesis is far removed from planning, and is not concerned with "the future" shown by planning scenarios, solving one of their problems could be thought of as having a template or schema with slots, values or functional elements to fill, a scenario could be used to show a series of "snapshots" of that template on its way to an attempt to instantiate the solution to the problem.

### 5.2.2 Use Of Scenarios In An Expert System

There are two levels to the question of using scenarios as the knowledge representation schema in an expert system. The first relates to the work done in this project to produce scenarios. The second level is deeper and involves the use of scenarios as the repository of the expert system's own domain knowledge.

Normally expert systems use a large body of rules to produce whatever output they generate. They can produce bacteria disease diagnoses or molecular models and there is no reason to belive they could not produce plans, abstract plans or scenarios. In fact, the set of heuristics used to organize the courses of action for the scenarios was basically a set of rules. Scenarios were used in a method of representing the knowledge for the user, not the expert system itself.

The deeper question of what form could the knowledge in an expert system could take, and could we use scenarios or some form of scenarios as a body of such knowledge will wait until later works when we examine the use of such a body of knowledge in a Tactical Assistant for a more complex domain.

### 5.2.3 Building Scenarios With Simulation

The form of simulation used in this Tactical Assistant is not exactly the same as normal simulation methods. Instead important events were chosen and courses of action built around them. There was no tree search or search of any kind as is normally associated with planning in complex domains.

The most important point of a tree-less simulation was that it allowed the system to avoid the processing involved in exploring multiple choice pathways. This was done by having more knowledge in the planning heuristics and their application.

The simulation technique used in this project is most similar to an event driven simulation as opposed to a discrete simulation. The difference is that in our method the temporal order of event posting and course of action organization is arbitrary and controlled by the goals given to the Tactical Assistant, rather than the processes in the operating in the world. Our viewpoint is "here is our plan to achieve these goals, what does that mean to the world?" while most simulations have the view "here is the world, what does that mean to any plans attempted?"

### 5.2.4 Use Of Symbolic Random Processes

The use of symbolic results from random processes comes from the same outlook expressed above. We decided that for most people planning errands it doesn't really matter if, for example, by going to a particular store at a particular hour one has an 'x' chance of being able to by a newspaper giving a utility of 'y' for that trip, rather what is important is the result: If they are sold out of newspapers you are going to have to go somewhere else to buy one. All the utility functions in the world don't help one bit when you are stuck in traffic and the movie starts in ten minutes.

So though our random processes are handled to some extent internally on a probabilistic basis - we have initial estimates and conditional modifiers; the important point is that if the estimated chances of the event happening exceed the level at which we wish to worry about such things then the event is shown to the user as having happened and its effects noted.

If the user decides such a level of concern is prudent then planning decisions can be made accordingly. Otherwise the posted effects can be ignored. Thus we feel that our symbolic presentation of the effects of random processes in a set of scenarios is more useful than in a numeric or utility function form.

## 5.3 Implications

Based on the results of this work we propose further examination of issues related to the four points pursued in this work.

First, we would like to continue the examination of the theory and use of scenarios. We would like to examine a much more complex domain, such as conflict simulation games. We feel this domain is more conducive to the kinds of knowledge intensive problem solving methods we have proposed than are games such as chess and offer a chance to really push our understanding of these methods. Such examination would hopefully further demonstrate the utility of the scenario concept as a problem solving tool.

Second, in building a Tactical Assistant for this new domain we would like to examine the problems and benefits of using a body of experience in the form of scenarios as the base of knowledge rather than a set of rules as in most expert systems.

Third, we would like to examine the methods of creating scenarios from a body of experience. What do you do if the experience does not exactly fit the current situation? Can we distill such methods of mapping from experience to reality?

Finally, as we were able to handle random processes in this work we would like to examine extending such techniques to handling inimical or unfriendly processes in the conflict simulation gaming domain. Can we use the fact that knowledge and goals are behind the actions of such processes to reduce our need to examine multiple futures?

## 6.0 References

[CHIE75] Chien, R. T. & Weissman S. "Planning & Execution in Incomletely Specified Environments" 4th IJCAI (1975) pp 169-174.

[DUPU79] Dupuy, Col. T. N. Numbers, Predictions & War Bobbs-Merrill Co. Inc. NY 1979.

[ESPO59] Esposito, V. West Point Atlas of American Wars Frederick A. Praeger Publishers, New York, 1959.

[FEIG77] Feigenbaum, E. A. "Themes and Case Studies in Knowledge Engineering" Stanford Tech Report STAN-CS-77-621 (1977).

[HAYE79] Hayes-Roth, B., et. al. "Modeling Planning as an Incremental Opportunistic Process" 6th IJCAI (1979) pp 375-383.

[HAYE80] Hayes-Roth, B. "Projecting the Future for Situation Assesment and Planning" Rand Note N-1600-AF Nov 1980.

[LOWR78] Lowrance J., Grasper Language Reference Manual COINS Technical Report 78-20, December 1978.

[McDE78] McDermott, D. "Planning & Acting" Cognitive Science 2 71-109 (1978).

[MINS75] Minsky, M. "A Framework for Representing Knowledge" In P. H. Winston (Ed.) The Psychology of Computer Vision, NY:McGraw-Hill 1975.

[NILS80] Nilsson, Nils Principles of Artificial Intelligence Tioga Publishing Co., Palo Alto 1980.

[PRAT81] Pratuch, T. "Advanced Tactics: Reality and Games" Moves nrs. 54, 55, 56; Dec/Jan, Feb/Mar, Apr/May 1981.

[SIKL74] Siklossy, L. & Dreussi, J. "Simulation of Executing Robots in Uncertain Environments" Presented at the National Computer Conference 1974.

[SLAT77] Slate, D. J. & Atkin, L. R. "Chess 4.5 - The Northwestern University Chess Program" In P. Frey (Ed.) Chess Skill in Man & Machine, Springer Verlag 1977. See also International Computer Chess Assoc. Newsletter of Nov 1980.

[SPRO77] Sproul, R. F. "Decision Theory and Artificial Intelligence II: The Hungry Monkey" Cognitive Science 1 158-192 (1977)

[USAR69] "Basic Staff Manual" FM-101-5; U. S. Army AG Publication Center Baltimore Md 1969.

[WANG80] Wang, D. Personal Communications, 1980.

[WESS77] Wesson, R. Air Traffic Control Ph. D. Thesis University of Texas Austin 1977.

# APPENDIX A

## Examples Of Experiments

This section presents some of the raw output from some of the experiments.

Experiment four was run with four goals. It had two sets of conflicting mandatory time goals, the first pair occuring early and the other later enough to avoid any time conflict.

The scenario produced shows the breaking of the main group of actions into four subchoices - one bifurcation caused by the early time conflict and the other by the later time conflict.

```
"The goals in these scenarios:"
((get-newspaper
 (NAME (VALUE (get-newspaper)))
 (DUE-DATE-OF-GOAL (VALUE ((d-mand 3))))
 (VALUE-OF-GOAL (VALUE (4)))
 (LOCATION-OF-GOAL (VALUE ((l-book x))))
 (TYPE-OF-GOAL-OBJECT (VALUE (i-paper)))
 (TYPE-OF-GOAL-VERB (VALUE (v-get)))
 (TYPE-OF-GOAL (VALUE (a-transact))))
(get-groceries
 (NAME (VALUE (get-groceries)))
 (DUE-DATE-OF-GOAL (VALUE ((d-mand 3))))
 (VALUE-OF-GOAL (VALUE (4)))
 (LOCATION-OF-GOAL (VALUE ((l-groc x))))
 (TYPE-OF-GOAL-OBJECT (VALUE (i-food)))
 (TYPE-OF-GOAL-VERB (VALUE (v-get)))
 (TYPE-OF-GOAL (VALUE (a-transact))))
(get-shoes
 (NAME (VALUE (get-shoes)))
 (DUE-DATE-OF-GOAL (VALUE ((d-mand 8))))
 (VALUE-OF-GOAL (VALUE (4)))
 (LOCATION-OF-GOAL (VALUE ((l-dept x))))
 (TYPE-OF-GOAL-OBJECT (VALUE (i-shoes)))
 (TYPE-OF-GOAL-VERB (VALUE (v-get)))
 (TYPE-OF-GOAL (VALUE (a-transact))))
(get-tools
 (NAME (VALUE (get-tools)))
 (DUE-DATE-OF-GOAL (VALUE ((d-mand 8))))
 (VALUE-OF-GOAL (VALUE (4)))
 (LOCATION-OF-GOAL (VALUE ((l-hdw x))))
 (TYPE-OF-GOAL-OBJECT (VALUE (i-tool)))
 (TYPE-OF-GOAL-VERB (VALUE (v-get)))
 (TYPE-OF-GOAL (VALUE (a-transact)))
)
 (Random Threshold 10000)

 (SCENARIO scenarioxaae)
 (ACTION first OCCURS AT s-home)
 (TIME 1)
 (ACTION moveaaa:goto-s-ss2 OCCURS AT s-ss2)
 (TIME 3)
 (ACTION actionxaah:get-groceries OCCURS AT s-ss2)
 (TIME 8)
 (ACTION actionxaaj:get-tools OCCURS AT s-ss2)

 (SCENARIO scenarioxaac)
 (ACTION first OCCURS AT s-home)
 (TIME 1)
 (ACTION movexaab:goto-s-ss2 OCCURS AT s-ss2)
 (TIME 3)
 (ACTION actionxaag:get-newspaper OCCURS AT s-ss2)
 (TIME 8)
 (ACTION actionxaaj:get-tools OCCURS AT s-ss2)

 (SCENARIO scenarioxaaf)
 (ACTION first OCCURS AT s-home)
 (TIME 1)
 (ACTION movexaac:goto-s-ss1 OCCURS AT s-ss1)
 (TIME 3)
 (ACTION actionxaah:get-groceries OCCURS AT s-ss1)
 (TIME 8)
 (ACTION actionxaai:get-shoes OCCURS AT s-ss1)

 (SCENARIO scenarioxaab)
 (ACTION first OCCURS AT s-home)
 (TIME 1)
 (ACTION movexaad:goto-s-ss1 OCCURS AT s-ss1)
 (TIME 3)
 (ACTION actionxaag:get-newspaper OCCURS AT s-ss1)
 (TIME 8)
 (ACTION actionxaai:get-shoes OCCURS AT s-ss1)
```

Experiment seven was run with all seven possible tasks requested. The system had to make travel decisions, timing decisions, etc. The results show the system in its full working glory, as well as the problems caused by strict observance of the heuristics as they were stated. To get a "more reasonable" plan would require either a different ordering of the heuristics or else a larger and more clever means of applying them.

```
"The goals in these scenarios:"
(cash-paycheck
   (NAME (VALUE (cash-paycheck)))
   (DUE-DATE-OF-GOAL (VALUE ((d-sug 10))))
   (VALUE-OF-GOAL (VALUE (10)))
   (LOCATION-OF-GOAL (VALUE ((l-bank x))))
   (TYPE-OF-GOAL-OBJECT (VALUE (i-money)))
   (TYPE-OF-GOAL-VERB (VALUE (v-get)))
   (TYPE-OF-GOAL (VALUE (a-transact))))
  (get-groceries
   (NAME (VALUE (get-groceries)))
   (DUE-DATE-OF-GOAL (VALUE ((d-sug 15))))
   (VALUE-OF-GOAL (VALUE (5)))
   (LOCATION-OF-GOAL (VALUE ((l-groc x))))
   (TYPE-OF-GOAL-OBJECT (VALUE (i-food)))
   (TYPE-OF-GOAL-VERB (VALUE (v-get)))
   (TYPE-OF-GOAL (VALUE (a-transact))))
  (get-newspaper
   (NAME (VALUE (get-newspaper)))
   (DUE-DATE-OF-GOAL (VALUE ((d-sug 3))))
   (VALUE-OF-GOAL (VALUE (5)))
   (LOCATION-OF-GOAL (VALUE ((l-book x))))
   (TYPE-OF-GOAL-OBJECT (VALUE (i-paper)))
   (TYPE-OF-GOAL-VERB (VALUE (v-get)))
   (TYPE-OF-GOAl (VALUE (a-transact))))
  (get-new-shoes
   (NAME (VALUE (get-new-shoes)))
   (DUE-DATE-OF-GOAL (VALUE ((d-sug 3))))
   (VALUE-OF-GOAL (VALUE (6)))
   (LOCATION-OF-GOAL (VALUE ((l-dept x))))
   (TYPE-OF-GOAL-OBJECT (VALUE (i-shoes)))
   (TYPE-OF-GOAL-VERB (VALUE (v-get)))
  (get-tools
   (NAME (VALUE (get-tools)))
   (DUE-DATE-OF-GOAL (VALUE ((d-sug 13))))
   (VALUE-OF-GOAL (VALUE (5)))
   (LOCATION-OF-GOAL (VALUE ((l-hdw x))))
   (TYPE-OF-GOAL-OBJECT (VALUE (i-tool)))
   (TYPE-OF-GOAL-VERB (VALUE (v-get)))
   (TYPE-OF-GOAL (VALUE (a-transact))))
  (get-birthday-present
   (NAME (VALUE (get-birthday-present)))
   (DUE-DATE-OF-GOAL (VALUE ((d-sug 13))))
   (VALUE-OF-GOAL (VALUE (5)))
   (LOCATION-OF-GOAL (VALUE ((l-toy x))))
   (TYPE-OF-GOAL-OBJECT (VALUE (i-toy)))
   (TYPE-OF-GOAL-VERB (VALUE (v-get)))
   (TYPE-OF-GOAL (VALUE (a-transact))))
  (go-to-movies
   (NAME (VALUE (go-to-movies)))
   (DUE-DATE-OF-GOAL (VALUE ((d-mand 20))))
   (VALUE-OF-GOAL (VALUE (8)))
   (LOCATION-OF-GOAL (VALUE ((l-movies x))))
   (TYPE-OF-GOAL-OBJECT (VALUE (i-movie)))
   (TYPE-OF-GOAL-VERB (VALUE (v-get)))
   (TYPE-OF-GOAL (VALUE (a-transact))))
 )
(Random Threshold 0)

(SCENARIO scenarioxaaa)
(ACTION first OCCURS AT s-home)
(TIME 1)
(ACTION movexaab:goto-s-ss2 OCCURS AT s-ss2)
```

```
("action:" (movexaab) "may be delayed due to heavy traffic")
(TIME 7)
(ACTION actionxaao:get-tools OCCURS AT s-ss2)
("action:" (get-tools) "may fail to achieve goal and no
 retry is possible")
(TIME 8)
(ACTION actionxaal:get-groceries OCCURS AT s-ss2)
("action:" (get-groceries) "may fail to achieve goal.  Retry
  should be made at locations:" (s-ss1))
(TIME 9)
(ACTION movexaac:goto-s-dtn OCCURS AT s-dtn)
("action:" (movexaac) "may be delayed due to heavy traffic")
(TIME 12)
(ACTION actionxaap:get-birthday-present OCCURS AT s-dtn)
("action:" (get-birthday-present) "may fail to achieve goal and no
 retry is possible")
(TIME 17)
(ACTION movexaaa:goto-s-mls OCCURS AT s-mls)
("action:" (movexaaa) "may be delayed due to heavy traffic")
(TIME 20)
(ACTION actionxaaq:go-to-movies OCCURS AT s-mls)
("action:" (go-to-movies) "may fail to achieve goal and no retry
 is possible")
(TIME 21)
(ACTION actionxaak:cash-paycheck OCCURS AT s-mls)
(TIME 22)
(ACTION actionxaam:get-newspaper OCCURS AT s-mls)
("action:" (get-newspaper) may fail to achieve goal.  Retry should
 be made at locations:" (s-ss1 s-ss2 s-dtn))
(TIME 23)
(ACTION actionxaan:get-new-shoes OCCURS AT s-mls)
("action:" (get-new-shoes) "may fail to achieve goal.  Retry should
 be made at locations:" (s-ss1 s-ss2))
```