Goal-directed Software Assistant for a Planning Advisory System

Peng Yu
Computer Science and Artificial Intelligence Lab, MIT
yupeng@mit.edu

MAS.761 Final Project Report

ABSTRACT

In this paper, I present an assistant program for a scientific plan advisory system that can help amateur users identify problems in unexpected situations and find related instructions for resolving them. This assistant program uses commonsense reasoning to map the users' problem descriptions into related concepts used by the advisory system, and to generate explanations for the possible causes of failure and instructions on recovery procedures using the system. The commonsense reasoning layer resolves the mismatch between the users' knowledge and the advisory system's model, and therefore enables the user to easily communicate to the system's high functionality interface.

INTRODUCTION

In today's software market, there are many professional applications developed for the specific domains of business, industry and everyday lives. Some widely used software in this category includes Photoshop for image editing, Solidworks for mechanical design, and Microsoft Project for project management. Their rich capabilities have greatly helped professionals in their specialties domain to address more problems and more efficiently. However, their complicate user interfaces have long been a barrier that prevents most amateur users to benefit from their capabilities. Usually, this type of software employs high functionality interfaces with countless buttons and menus to make it more efficient for advanced users to access desired functions quickly. In addition, their functions and operations are often described using non-intuitive expressions for improved accuracy, which creates many jargons and makes it even more difficult for beginners.

I summarize the difficulties of using a high functionality interface as the following two issues:

(1) Communicating accurate problem descriptions or expected outcomes.

(2) Identifying the correct procedures for achieving the objectives.

Much effort has been devoted into (2) through developing tutorials and training sessions for beginners to learn the operations. Nowadays, it is very easy to find detailed step-by-step instructions for any software online or in their help menu. On the other hand, little has been done for the first issue on improving the communication between users and high functionality software. Modern software is still using indexed list or searching help contents to locate the answers to the users' questions, which were developed decades ago. The methods assume that the users have prior knowledge about the software and can describe the problems accurately, which is not a realistic expectation for beginner users.

In this paper, a new approach is presented to bridge this gap by incorporating a commonsense reasoning layer to map the users' problem descriptions into the definitions used by the software and its tutorials. Given a problem description, I first generate explanations of the problem using similar concepts in the software's knowledge base. Then explore possible causes and effects by propagating neighboring assertions with causal relations. Finally, the solutions to the users' problem are generated by looking through related assertions to all concepts found in the first two steps with resolution relations. This is motivated by the goal-directed user interface presented in [2], with application in the domain of high-functionality software.

This approach has been implemented as an assistant program for a cruise mission plan advisory system. The plan advisory system is used to help oceanographers arrange and schedule tasks during a scientific expedition [1]. The interface of the plan advisor is one example of high functionality user interfaces with more than 200 operations, most of which are non-trivial for ocean scientists without background in automated planning. The assistant program, named as "Captain Kirk", can automatically generate instructions that address the users' problems. The user can express the problem or describe the faulty situation in plain English, without having any prior knowledge or reference to the command dictionary.

In this paper, I first present the background of the plan advisory system and formally define the problem of identifying and mapping users' problem description to instructions. Then I present the detailed approach and implementation of the assistant program. Finally, I demonstrate its application using a failed plan recovery scenario, and show a set of initial experiment results.

BACKGROUND

The cruise plan advisory system was developed jointly with the Deep Submergence Lab at Woods Hole Oceanographic Institute, and is aimed at providing decision support for scientists in mission planning before and during an expedition. An expedition cruise can lasts for two to five weeks, and is usually supporting multiple scientific experiments at different locations. Planning such a cruise is very challenging due to its complexity and uncertainty: there are always more tasks to do than what the cruise can support, and there can be large uncertainty in getting the expected science returns. In addition, unexpected weather changes and equipment failures are often encountered at sea, which may interrupt the execution of cruise plans and delay all operations for days. Therefore, it takes years of experience for a scientist to be qualified for leading an expedition. A lead scientist must learn to make the cruise plan as robust as possible and have a prioritized list of scientific goals. When unexpected failure occurs and makes the original plan over-subscribed, he/she must make trade-offs between objectives and adjust the plan accordingly to ensure the completion of high priority tasks.

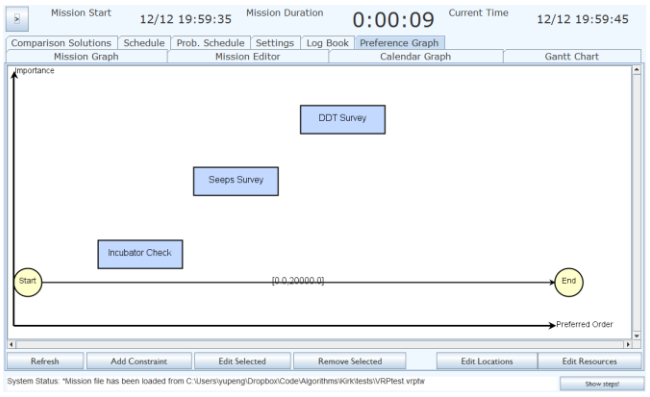
The plan advisory system was designed to take over some of the planning tasks, reduce the workload of the scientists and increase the reliability of the cruise plans. It provides the following three capabilities:

1. Task selection, sequencing and scheduling.
2. Failure detection and recovery through goal relaxation.
3. Human resources and assets management.

The advisory system is supported by the Conflict-Directed Relaxation with Uncertainty algorithm. The algorithm was developed for efficiently resolving infeasible conditional problems. With the advisory system, the users only need to provide the goals of the expedition, temporal requirements and uncertainty in each activity. The system will provide a solution, which consists of the following four items:

1. A sequence of activities that achieves the goals.
2. The schedule of each activity.
3. (Optional) Low priority goals that cannot be achieved.
4. (Optional) Temporal requirements that have to be relaxed.

The user specifies the goals and requirements using a 2D graphical interface (Figure 1). The solutions are presented in both a calendar like graph (Figure 2) and a Gantt chart (Figure 3). The users control the solution process using a separate control panel. They can also edit the generated plan directly, or recovery from a failure using the interruption handler (Figure 4). The advisory system was implemented to reduce the layers of menus and provide more direct access to functions. Therefore, there are considerable amount of buttons on each tab and each responsible for a specialized function.



System Status: *Mission file has been loaded from C:\Users\yupeng\Dropbox\Code\Algorithms\Kirk\tests\VRPtest.vrptw

Figure 1 Interactive Goal Graph

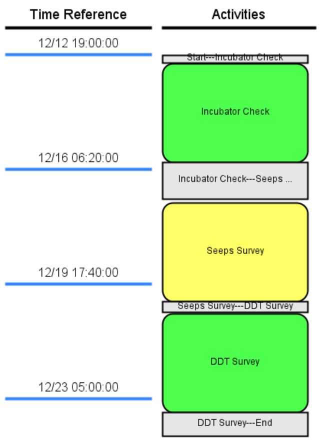
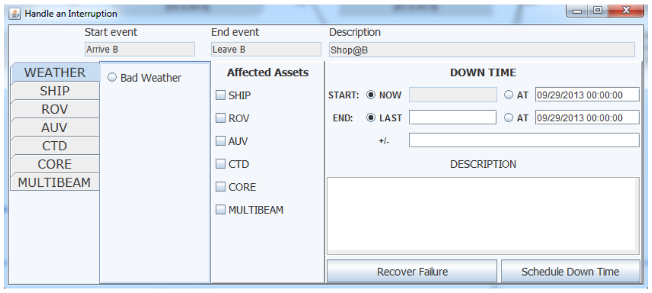


Figure 2 Calendar Like Solution Graph

Figure 3 Gantt Chart Like Solution Graph

The initial feedback returned from the users was mixed: the scientists believe that the capabilities provided by the advisory system would be very useful on cruise planning, but the user interface is too complicated to learn. Most of them are geologists and do not have background in computer science and artificial intelligence. They find it hard to understand the functions and descriptions of each button. It is also difficult for the scientists to communicate the problems to the system, and identify the correct functions to use for resolving them.

The feedback revealed two issues. First, there is a mismatch between the scientists' and the system's problem descriptions. This is due to the gap between their knowledge base and contributes to the communication problem.



Handle an Interruption

Start event: Arrive B
End event: Leave B
Description: Shop@B

START: [✓ NOW] AT 09/29/2013 00:00:00
END: [✓ LAST] AT 09/29/2013 00:00:00

Figure 4 Failure Recover Scheduler Window

Second, the layout of the interface is not intuitive enough for the users to find the correct procedures to address their problems. The second problem has been well studied and is usually addressed by making tutorials and instruction sets for each specific problem. However, there is not much work done for the first problem: people are still using keyword search or index to locate the answers to their problems. If the users do not have much prior knowledge about the system and some terminologies, it will be very difficult for them to communicate their problem accurately using these methods. In this paper, I address this problem by using the common-sense reasoning tool, D4D, to find the mapping between the users' and system's descriptions.

PROBLEM STATEMENT

In this section, I define the problem addressed in this paper: mapping users' goal/failure descriptions to the advisory system's step-by-step instructions that help the users achieve their goals/resolve their problems. I will also demonstrate the inputs and expected outcomes of the system.

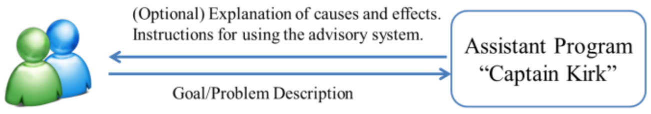

(Optional) Explanation of causes and effects.
Instructions for using the advisory system.
Goal/Problem Description
Assistant Program
"Captain Kirk"

Figure 5 Expected Behavior of the Assistant Program

The flow of the assistant program is presented in Figure 5. The program has a text box for the user to type or speak the input. The input can be a plain English sentence that describes the user's objective, such as "I want to extend the exploration time at site A". It can also be a short expression that describes a (potential) failure situation, such as "propeller shaft overheating". The users are not required to use any specific terminology or reference the command dictionary in the descriptions.

The advisory system should give two outputs:

1. (Optional) If the user is looking for a solution to a problem, the advisory system should generate explanations on the possible causes and effects of the problematic situation.
2. Step-by-step instructions for the user to achieve the goal or resolve the problem. There might be more than one set of instructions if there are multiple problems need to be addressed.

For example, the user may tell the assistant program about an expected weather change: "Typhoon is expected in the next 24 hours". The assistant program should provide the following explanations and instructions:

- Typhoon is a type of bad weather, and can cause ship down time and sensor down time.
- Please use the ship and sensor down time scheduler to adjust your plan, following these steps:
  (1) double-click the current activity to bring up the down time scheduler window; (2) select Ship and Sensor as the affected assets; (3) Input the expected start time and duration of the bad weather; (4) click "schedule down time" button to confirm and close the window.

APPROACH

A 5-step approach is constructed to communicate the users' descriptions to the advisory system, and generate the correct instructions for addressing the users' needs. The commonsense reasoning capability is provided by D4D [3] over two knowledge bases: one customized knowledge base, called Kirk, for concepts related marine time activities and the advisory system; the other one is blended using Kirk and ConceptNet 4. The five steps in this approach are the followings:

1. Find the most similar known concepts.
2. Find explanation and definition concepts.
3. Identify causes and effects.
4. Generate abstract solutions for goals or problems.
5. Map the abstract solutions to detailed instructions.

Step 1 and 2 are used to *explain* the users' requests, through finding concepts and related definitions in the specialized knowledge that are most similar to the users' descriptions. Step 3 *explores* possible causes and effects, if the user describes a problem or failure situation. Finally, Step 4 and 5 *generates* instructions for the users, based on the explanations and consequences discovered in the previous steps. This "Explain – Explore – Generate" process is illustrated in Figure 6.

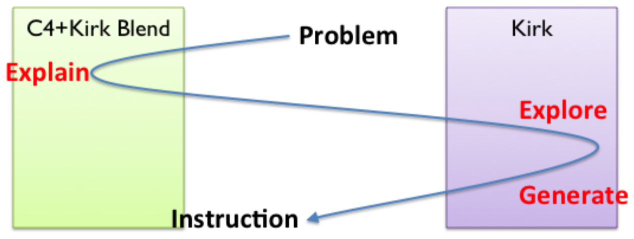
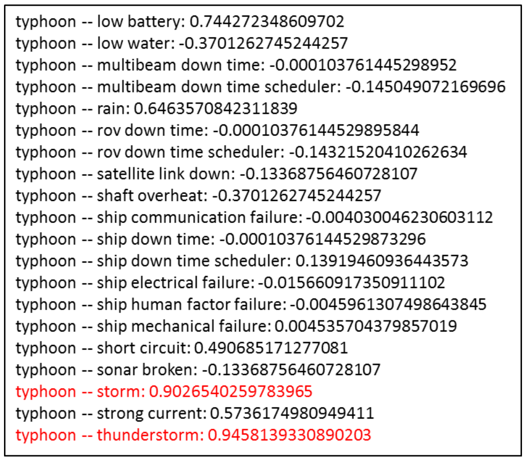


Figure 6 An Overview of the Approach

Explain

The users' descriptions are usually in plain English and may not direct match with the terminologies used in the Kirk knowledge base. The first step is to build the connection between them. To map the request from the commonsense domain to the specialized domain about science explorations, the assistant program iterates through all concepts in Kirk and asks ConceptNet 4 to provide a score on their similarities to the concepts in the users' descriptions. The top two concepts are selected as the explanation for the users' description in the specialized domain.

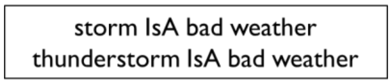
For example, if the user asks about "typhoon", the concepts "storm" and "thunderstorm" in Kirk will be selected since they the highest similarity score (Figure 7). The rest of the concepts are discarded to restrict the breadth of the mapping and simplify the results.



```
typhoon -- low battery: 0.744272348609702
typhoon -- low water: -0.3701262745244257
typhoon -- multibeam down time: -0.00010376144529852
typhoon -- multibeam down time scheduler: -0.145049072169696
typhoon -- rain: 0.6463570842311839
typhoon -- rov down time: -0.00010376144529895844
typhoon -- rov down time scheduler: -0.14321520410262634
typhoon -- satellite link down: -0.13368756460728107
typhoon -- shaft overheat: -0.3701262745244257
typhoon -- ship communication failure: -0.004030046230603112
typhoon -- ship down time: -0.00010376144529873296
typhoon -- ship down time scheduler: 0.13919460936443573
typhoon -- ship electrical failure: -0.015660917350911102
typhoon -- ship human factor failure: -0.0045961307498643845
typhoon -- ship mechanical failure: 0.004535704379857019
typhoon -- short circuit: 0.490685171277081
typhoon -- sonar broken: -0.13368756460728107
typhoon -- storm: 0.9026540259783965
typhoon -- strong current: 0.5736174980949411
typhoon -- thunderstorm: 0.9458139330890203
```

Figure 7 Mapping "Typhoon" to the Kirk Knowledge Base

Next, the set of explanation concepts are expanded for one more step using definitive and related concepts. This is done through exploring their neighboring assertions with "IsA", "DefinedAs" and "ConceptuallyRelatedTo" relations. For example, concepts "storm" and "thunderstorm" can be extended to a more general concept "bad weather" through assertions "storm is a bad weather" and "thunderstorm is a bad weather" (Figure 8).

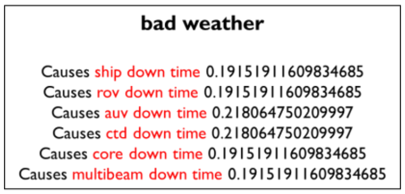


storm IsA bad weather
thunderstorm IsA bad weather

Figure 8 Extending Concepts using Definitions

Explore

The sets of instructions for the advisory system are designed to resolve specific issues. However, the user may not be aware of the real problems and provide accurate descriptions at all time. Instead, the descriptions may be a simple statement on their observations, such as "the shaft is overheating". In the next stage, the assistant program explores the possible causes and consequences of the problem/failure stated by the users. This is achieved through examining the neighboring assertions of all related concepts generated in the previous two steps: if the assertion has a causal relation, such as "Causes", "CreatedBy" and "HasA", then its cause or effect concept will be recorded.



bad weather

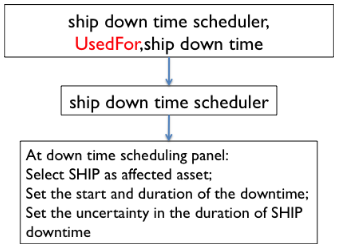
```
Causes ship down time 0.191519116009834685
Causes rov down time 0.191519116009834685
Causes auv down time 0.218064750209997
Causes ctd down time 0.218064750209997
Causes core down time 0.191519116009834685
Causes multibeam down time 0.191519116009834685
```

Figure 9 Exploring Causes and Effects

For example, given the concepts "storm", "thunderstorm" and "bad weather", the following six consequences can be found (Figure 8). They are ranked based on the truth score given by the blended Kirk and ConceptNet 4 knowledge base. This score can also be used to indicate the priority of each effect: if the user does not have enough time to address all the problems, he/she can start with the problems on top of the list.

Generate

The final stage of the mapping process is to locate the solutions, which are sets of instructions that achieve the users' goals or resolve their problems. In the Kirk knowledge base, each instruction set is represented by an abstract concept and is linked to other concept through solution relations, such as "UsedFor" and "CapableOf". These abstract solution concepts are the keys to the complete sets of instructions, and are stored in a pre-generated hash table. To locate these abstract concepts once the causes and effects of the user's problem has been identified in the previous steps, the assistant program searches through the neighboring assertions with solution relations. The last step is to map the abstract solution concepts to the complete instructions, and present them to the user.



ship down time scheduler,
UsedFor,ship down time

ship down time scheduler

At down time scheduling panel:
Select SHIP as affected asset;
Set the start and duration of the downtime;
Set the uncertainty in the duration of SHIP downtime

Figure 10 Generating Solution Concepts and Instructions

For example, given the failure "ship down time", the assistant program will identify the assertion "ship down time scheduler UsedFor ship down time". It then searches through the hash table using the key "ship down time", and locate the corresponding instructions for the user (Figure 10).

EVALUATION

In this section, I evaluate the performance of this assistant program in helping users find the instructions that resolve their problems. The performance has two aspects: accuracy and coverage. Accuracy describes if the assistant program finds the correct instructions for the problems, while coverage describes how many types of situations the assistant program can handle. A set of twenty different user requests, presented in plain English, is used in this evaluation. These sample requests come from some common failures during a science expedition cruise, such as “we are very sleepy” and “engine temperature high”. Due to time limit, this evaluation only covers the requests in the domain of failure recovery. Requests in other domains, such as model modifications and resource management, will be tested in future evaluations. The results are presented in Table 1.

Table 1 Summary of Evaluation Results

	Instructions Found		Instructions Not Found
	Accurately Resolved	Inaccurately Resolved	
storm is coming	1	0	0
we are sleepy	1	0	0
shaft overheat	1	0	0
engine temperature high	0	1	0
typhoon is coming	1	0	0
communication is offline	0	1	0
sonar is broken	0	1	0
satellite link is down	0	1	0
air leak detected	0	1	0
high wave is expected	1	0	0
water supply insufficient	0	0	1
fuel is running low	0	1	0
power outage detected	0	1	0
low battery power	1	0	0
navigation system offline	1	0	0
food supply is low	0	1	0
internet connection lost	0	1	0
crew fatigue level high	1	0	0
short circuit is detected	1	0	0
gas pressure is high	0	1	0
Total	9	10	1

As can be seen from the table, the assistant program found instructions for more than 90% of the requests. This is mainly due to the wide coverage of ConceptNet 4. It made several intelligent inferences for connecting unknown concepts to the Kirk knowledge base, such as “Navigation – satellite link” and “sleepy -- fatigue”. However, less than 50% of the requests were answered with the correct instructions: the assistant program has a large bias towards connecting failures to concepts related bad weathers, such as “hurricane” and “thunderstorm”. It is more likely to assign a high similarity score to those weather related concepts than others, hence causes the inaccuracy problem.

To address the issue and improve the performance of the assistant program, it is necessary to refine the inference process of D4D and make it more reliable. Currently its similarity score may be inconsistent between very similar concepts, especially on blended knowledge bases. In addition, sometimes it is useful to allow the user adjust the trade-off between accuracy and coverage: when the user has enough time he/she may want to see as many options as possible and make the decision themselves. On the other hand, if the situation is urgent, the user may only need to instruction with top priority and execute it immediately.

CONCLUSION

This paper introduces the design and implementation of an assistant program, called “Captain Kirk”, for a mission advisory system that helps marine scientists to plan their expeditions. The assistant program helps mapping the users’ problem and goal descriptions, presented in plain English, to sets of pre-defined instructions. This makes it much easier for beginner users without computer science background to utilize the capabilities of the advisory system, which comes with a high functionality interface. Preliminary evaluation results have demonstrated that the assistant program covers a wide range of requests. The evaluation also reveals its lack of accuracy in the identification of relevant instruction, which will be addressed in future development.

REFERENCES

[1] Continuously Relaxing Over-constrained Conditional Temporal Problems through Generalized Conflict Learning and Resolution, Peng Yu and Brian Williams, In Proceedings of the Twenty-third International Joint Conference on Artificial Intelligence (IJCAI-2013), August 2013.

[2] A goal-oriented interface to consumer electronics using planning and commonsense reasoning, Henry Lieberman and José Espinosa, In Proceedings of the 11th international conference on Intelligent user interfaces, pages 226–233, New York, NY, USA, 2006. ACM Press.

[3] Commonsense reasoning in and over natural language, Hugo Liu and Singh P., In Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES-2004), Brighton, U.K, 2004.