MODERN TRENDS IN INFORMATION TECHNOLOGY
PVS Rao, P Sadanandan (Editors)
Tata McGraw-Hill Publishing Company Ltd

Discussing, Using and Recognizing Plans in SNePS Preliminary Report - SNACTor : An Acting System

Deepak Kumar, Syed S. Ali, Stuart C. Shapiro
State University of New York at Buffalo, USA

Abstract

This project involves designing a system for discussing, using, and recognizing plans using a semantic network knowledge representation paradigm. We consider plans as mental objects that can be recognized and inferred from natural language dialog in a specific domain. Plans are represented and generated in a planning formalism using SNePS (the Semantic Network Processing System). In this primilinary report we describe SNaCTor (SNePS Actor) an extension to SNePS, designed for generating and acting plans. The planning formalism used is loosely based on the GRAPPALE Plan Formalism (GPF) developed at the University of Massachusetts.

1. Introduction

This project involves designing a system for discussing, using, and recognizing plans in a semantic network knowledge representation paradigm. The objectives of this work are:

(1) Design a representation for plans and rules for reasoning about plans within an established knowledge representation/reasoning (KRR) system; enhance the KRR system so that it can act according to such plans;

(2) Write a grammar to direct an established natural language processing (NLP) system to analyze English sentences about plans and represent the semantic/conceptual content of the sentences in the representation designed for objective (1); the resulting NLP system should be able to: accept sentences describing plans, and add the plans to its "plan library"; answer questions about the plans in its plan library; accept sentences describing the actions of others, and recognize when those actions constitute the carrying out of a plan in its plan library.

The KRR system being used is SNePS [SHP79], and the NLP system to be modified for this purpose is CASSIE[SHP87]. This project is being carried out jointly by teams at the University of Massachusetts (UMass) and SUNY at Buffalo (UB). The UB group is responsible for enhancing SNePS/CASSIE according to the objectives listed above. The UMass group is responsible for test-

ing the enhanced system in the specific domains of the Blocks World, tutoring, and space launch narratives.

In this report we describe our knowledge representation scheme and the SNePS Actor (SNACTor) an extension to SNePS, designed for acting out plans generated by the planning formalism. The planning formalism used is loosely based on the GRAPPLE Plan Formalism (GPF) [HUF87].

2. Knowledge Representation

The design of our representations is based on a theory of intensional propositional knowledge representation in SNePS [SHP87]. It is a major hypothesis of the current project that plans are mental objects that can be represented in such a formalism. We can discuss plans with each other, reason about them, formulate them, execute them, and recognize when others seem to be following them. An AI system, using SNePS as a basis for its belief structure, should be able to do these things.

Plans, being structures of actions, states, and other plans, reesemble reasoning rules, which are structures of beliefs. However, they are different in important ways: reasoning rules are rules for forming new beliefs based on old beliefs, plans are rules for acting; a belief, once formed, need not be formed again, an action may need to be performed multiple times; the temporal order of assessing old beliefs and forming new beliefs is flexible, the temporal order of performing actions is crucial to the correct carrying out of a plan. The representation of reasoning rules in SNePS, and the algorithm for reasoning according to them, implemeted in SNIP - SNePS Inference Package, have been carefully designed to make reasoning flexible and efficient. In this project, we are undertaking a similar design of the representation and use of plans. The resulting system called SNACTor (for SNePS ACTor) is described here.

2.1 The Representation of Acts

Our representation of acts is based on that of [ALM87]. He distinguishes the nodes for an act, the event of which act's being performed by a particular actor at a particular time, and the proposition of that event's having occurred. The benefit of this distinction is that the same node may be used to represent the same act (as required by the Uniqueness Principle [SHP79]), no matter who performs it, and no matter when performed. Figure 1 shows our representation of an act as node with an ACTION arc to a node that represents the action, and $OBJECT_1$, ...,$OBJECT_N$ arcs to the required objects of the action. For example, the SNePSUL (the SNePS command interpreter language) command for building a node representing the act of saying "FOO" is:

```
(build action say object1 FOO)
```

It is necessary to distinguish between primitive and complex (non-prmitive) actions. A primitive action is one that the system is capable of performing. The system is not capable of performing a complex action, so to carry out a complex action, the system must decompose it into a structure of other actions, which, eventually, must be primitive. We assert that an action is primitive using the standard SNePS MEMBER-CLASS case frame [SHP87]. To assert that "saying" is primitive, we execute the SNePSUL command:

```
(build member say class primitive)
```

Primitive actions may be supplied to the system by programming them in Lisp. One simple primitive action we have written is *say* -- the action of printing something on the output device, used as an example above. The code defining say is:

```
(defun say (n)
 (format t"~&~A~%"
 (first-atom (pathget n'object1))))
```

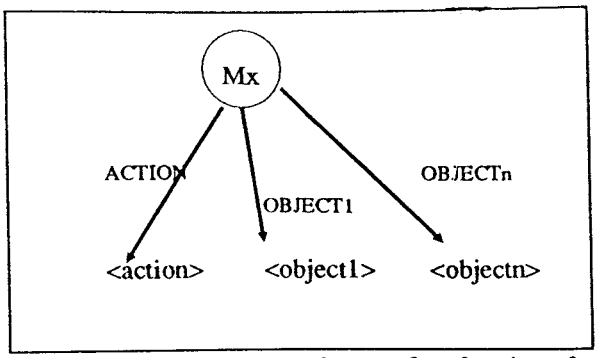

Figure 1. Mx Represents the act of performing of <action>

Several things may be noted about this definition: the action takes an act node with itself as action as its argument so the function call of acts is consistent regardless of the number of arguments the action takes; pathget takes a node and a path of arc label, and returns the list of nodes at the end of the given path from the given node; first-atom returns the first node in the list under the assumption that each argument of a say act will be a single node. In the remainder of this report, we will refrain from showing Lisp code of primitive acts, but will describe them in the format:

(say <node>) Primitive-action
"Prints <node> on the output device"

The arguments will be shown in a way that suggest their types.

We distinguish between actions that the system performs on the world (simulated by printing onto the output device), such as *pickup* and *putdown* (in the Blocksworld), and internal, mental actions, such as *believe*. The acting algorithm schedules the believing of the effects of acts after performing each act. Thus the *believe* primitive action looks like:

(believe <proposition>) Primitive-action
"causes the system to believe <proposition>, and removes from the system
any belief in <proposition's> negation."

We also have several "intention forming" actions whose effects are to schedule the execution of acts, modeling a cognitive agent forming the intention of performing that act. For example, a cognitive agent may intend to achieve a goal. For that purpose, we have the *achieve* action:

(achieve <proposition> Primitive-action
"Finds or infers a plan for achieving a state in which the <proposition>
holds, and schedules the carrying out of that plan."

Some other intention-forming actions crucial to our system are control actions. Control actions allow us to represent and carry out structured plans. Our repertoire of control actions consists of *snsequence, snif, select*, and *with*. *Snsequence* is the primitive action of performing two sub-acts in sequence:

(Snsequence <act1> <act2>) Primitive-action
"Schedules the execution of <act1> followed by <act2>"

Since either or both sub-acts can themselves be *snsequence* acts, we have a general structure for plans of sequential actions. *Snif* is a conditional branching action:

(snif <condition> <act1> <act2>) Primitive-action
"<condition> is a proposition. <act1> is scheduled
if the \,condition>
is believed to be true, <act2> is scheduled otherwise"

Select can be used to specify alternate actions towards the fulfillment of a particular goal:

(select <act1> <act2>... <actN>) Primitive-action
"<act1> ... <actN> are alternative acts that may achieve the same goal.
<act1> is scheduled first and if it fails, other alternatives are tried"

The *with* control action has a sensory component that probes the world (external or internal) to determine the objects of its action. For example, to remove a stack of blocks from the top of a given block in a tower, we need to look at the structure to determine the top block, before we can remove it from the top. The syntax of *with* looks like:

(with <probe-act> <do-act>) Primitive-action
"The objects on which the <do-acts> is to be
performed are determined by the <probe-act>"

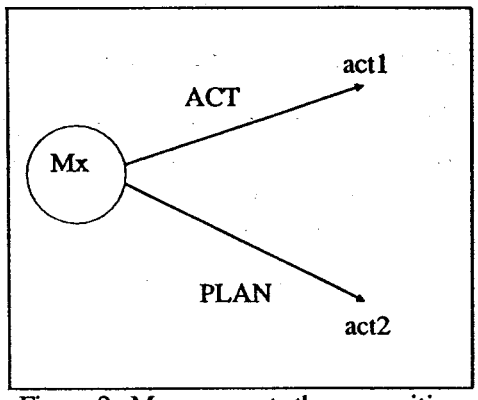

Figure 2. Mx represents the proposition that act2 constitutes a plan for carrying out act1. Act2 must be more decomposed than act1.

2.2 The Representation of Plans

A complete action is performed by decomposing it into a structure of simpler actions, which constitute a plan for carrying out the complex action. Our representation presupposes the hypothesis that this decomposition must take the objects of the action into account, but need not take the actor into account. Fig. 2 shows the decomposition relation that holds between two acts. The act at the end of the PLAN arc must be more decomposed than the act at the end of the ACT arc. e.g. suppose act $_1$ is complex, while act$_2$ is primitive. (We say that an act is primitive or complex just when its action is.) We can use a SNePS rule to assert that the plan for "asserting" anything is to say it by executing:

(build avb $x)
act (build action assert object1 *x)
plan (build action say object1 *x)

This kind of plan is one for carrying out a complex action. Another kind of plan is one for achieving some state of affairs. The representation of that kind of plan is shown in Fig. 3. For the representations of propositions, we use the

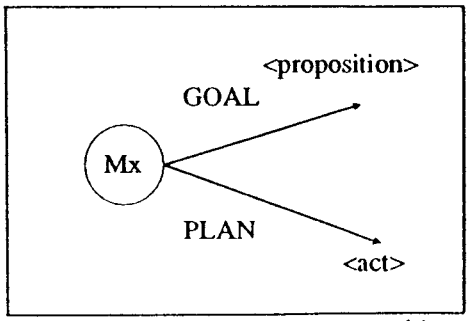

Figure 3. Mx represents the proposition that <act> constitutes a plan for achieving a state in which <proposition> is true

constructs shown in this report and those shown in [SHP87].

The system may already know a plan for a goal or complex act. However, if it does not already have such a plan, it may try to produce it by reasoning about the complex act or goal, effects of acts, etc. Such reasoning constitutes the planning activity. Effects of an act may be asserted into the SNePS network just like any other beliefs. Initially, we use the representation shown in Fig. 4, although it is certainly true that the effects of an act may depend on the actor, so this representation is too simplistic. Another simplifying assumption we are making is that all effects of a performed act occur.

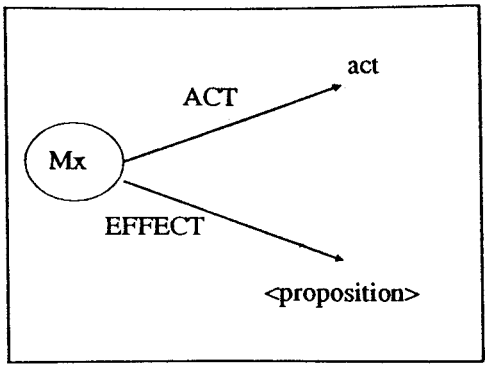

Figure 4. Mx represents the proposition that the effect of performing the <act> is that the <proposition> becomes true

3. SNACTOR : The SNePS Acting System

The acting system is composed of an acting executive (called "snact) and a queue of acts to be carried out. The lop-level algorithm is:

```
SNACT(QUEUE) ::=
REPEAT
        remove FIRST-ACT from QUEUE
        IF FIRST-ACT is primitive THEN
                DO FIRST-ACT
                INFER effects of FIRST-ACT,
                and schedule the believing of them
        ELSE {FIRST-ACT is complex}
                DEDUCE plans for carrying out FIRST-ACT,
                CHOOSE an appropriate plan and add it to QUEUE
        ENDIF
UNTIL QUEUE is empty.
```

Since more than one plan may be found, we have defined *a choose-plan* operation. *Choose-plan* can be made to use heuristics.

4. The NLP System

The natural language processing (NLP) system is a well- established augmented transition network (ATN) parser [SHP82]. A grammar for language associated with planning in Blocksworld (based on a test suite demo and the existing grammar) is in the initial stages of development.

Currently, the SNACTor system interacts with the user with SNePSUL commands. The intent of the plan grammar is to provide a "front-end" for discussing, using and recognizing plans in English which will generate the appropriate SNePSUL commands transparently to the user.

5. Conclusions

Our representation distinguishes actions, acts, propositions, decomposition (act-based) plans, and goal-oriented (state-based) plans. It also distinguishes between primitive actions/acts and complex actions/acts. We have actions that the system performs on the world, and internal, mental actions. We also have intention-forming actions, such as achieve, whose effects are to place acts on the act queue. The intention- forming actions snsequence, snif, select and with are control actions that permit us to represent and perform structured plans.

Our next tasks are clear. We will change the current CASSIE grammar so that conversations can be carried out in English instead of in SNePSUL. We will change our representation of plans so that advanced planning can be carried out. We will begin investigating the use of these plans to recognize when other actors are carrying out plans the system knows about. We will begin representing plans from the tutoring domain.

References

[ALM87] Almeida, M.J. **Reasoning about the Temporal Structure of Narratives**, Tech. Rep. No. 87-10, Dept. of Computer Science, SUNY at Buffalo, NY, 1987

[HUF87] Huffd, K.E. & Lesser, V.R. **The GRAPPLE Plan Formalism**, COINS Technical Report 87-08, Dept. of Computer and Information Sc., U. Mass, Amherst, MA, 1987

[SHP79] Shapiro, S.C. **The SNePS semantic network processing system**, In N.V. Findler, ed. Associative Networks: The Representation and use of Knowledge by computers, Academic Press, New York, 1979, 179-203

[SHP82] Shapiro, S.C. **Generalized augmented transition network grammars for generation from semantic networks**, Americal J. of Computational Linguistics 8, 1 (January-March 1982), 12-25

[SHP87] Shapiro, S.C. & Rapaport, W.J. **SNePS considered as a fully intensional propositional semantic network**, In G. McCalla & N. Cercone, eds. The Knowledge Frontier: Essays in the Representation of Knowledge, Springer-Verlag, New York, 1987, 262-315