UBGCCS-90: Proceedings of the Fifth Annual University at Buffalo Graduate-Conference on Computer Science

Keith E. Bettinger and Geetha Srikantan (eds.)

Department of Computer Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, NY 14260

UBGCCS-90 Organizing Committee:

Shriram Revankar, Chairman
Keith E. Bettinger
Peter B. Cullen
Lynda Spahr
Geetha Srikantan
Norman Stewart

The SNePS Acting System

Deepak Kumar, Syed S. Ali, Juergen Haas, and Stuart C. Shapiro
Department of Computer Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, NY 14260
kumard, syali, haas, shapiro@cs.buffalo.edu

Abstract

In UBGCCS-88, we presented propositional network representations for plans and acts to be used in the design of cognitive agent that can discuss, understand, and use its representations. The current version of the SNePS acting system can understand natural language utterances about a blocksworld. In this paper we present the architecture of this system. It includes the natural language understanding and generation components, the SNePS representations, the acting executive, the SNePS inference package, and the SNePS Belief Revision system. Several results obtained from the design and use of each of these components will be presented. Also, we will discuss specific research issues exposed by this study that we are currently working on.

1 Introduction

In [2, 4] we presented propositional network representations for plans and acts to be used in the design of cognitive agent that can discuss, understand, and use its representations. The current version of the SNePS acting system can understand natural language utterances about a blocksworld. In this paper, we present the architecture of this system. We present our representations of plans and acts, the design of the acting executive, how we make use of a belief revision system, and the design of the natural language component. This is followed by a brief look at how the system is able to discuss, use, and recognize plans in a blocksworld domain. Lastly, we present several future research issues related to this work that we are currently working on.

2 Architecture of the system

The architecture of the SNePS actor is as shown in Figure 1. The SNePS actor operates in a world inhabited by itself (*i.e.*, a single-agent world). The agent has beliefs that are stored as SNePS propositions in the agent's belief space (called a SNeBR context, see [7]). SNeBR (the SNePS system for Belief Revision), an assumption-based truth maintenance system [6, 5, 7], ensures that the agent's belief space is always consistent[1]. All interaction with the agent is done using the natural language component. Sentences are parsed by a grammar (written in ATN) and translated into SNePSUL (the SNePS User Language) commands and form beliefs in the agent's belief space. World model rules for reasoning in the agent's belief space are also translated and represented as agent's beliefs. An inference rule in SNePS is a structured proposition node of the form[2]

[1] During the course of acting, beliefs are removed and added. This is done using SNeBR operations. For example, one of the things SNeBR takes care of is when a belief is removed as a consequence of performing an action, all propositions derived using that belief are also removed.

[2] This linear representation of SNePS rules is designed to facilitate our current discussion. In SNePS rules one can have universal, existential, and numerical quantifiers over variables. The connectives available are and-entailment,

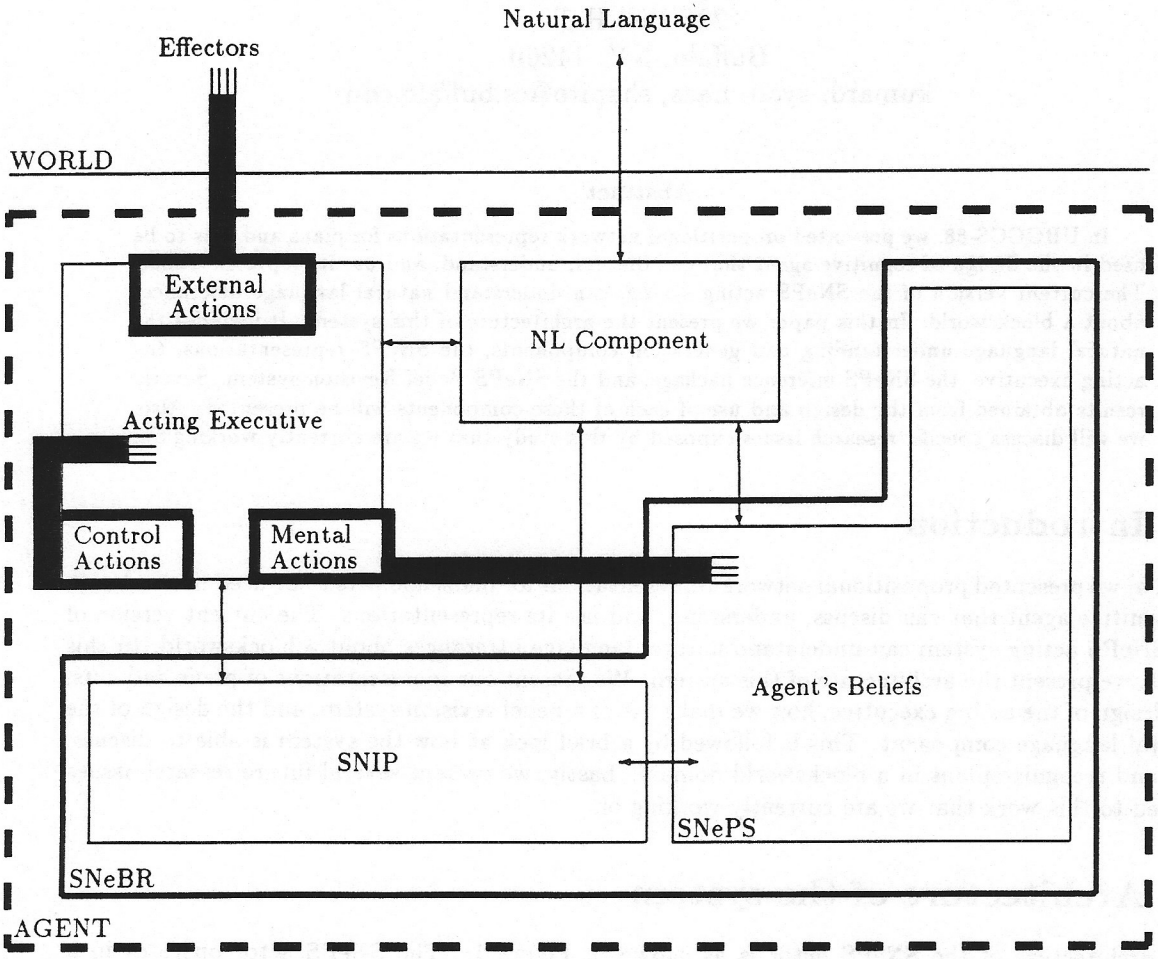

Figure 1: Architecture of The SNePS acting system

`AntCq(<ant-beliefs>, <cq-beliefs>)`

Roughly, the above rule is a specification of antecedent and consequent beliefs using appropriate quantifiers and connectives. SNIP can do forward, backward, or bidirectional inference using the same set of rules.

2.1 Representations for plans and acts

We treat acts/plans as mental objects. This enables the agent to discuss, formulate, use, recognize, and reason about acts/plans. This treatment is a significant advance over operator-based descriptions of plans. Operator-based formulations of actions tend to alienate the discussion of operators themselves since operators are usually specified in a different language from that used for representing beliefs about states. Moreover, plans (or procedural networks) constructed from these operators can only be accessed by specialized programs (critics, executors) and, like operators, are represented in still another formalism. Our representations for acts, goals, and plans build upon and add to the intensional propositional representations of SNePS. This framework enables us to tackle various tasks mentioned above in a uniform and coherent fashion.

We classify actions as being *external*—that affect the outside world, *control*—that affect the acting executive, and *mental*—that affect the set of beliefs. Plans (or complex acts) are represented as structured nodes comprising a set of external and control actions. Decompositions of plans/goals are specified using the following predicates[3]

```
PlanGoal( <some-plan>, <some-goal>)
PlanAct( <some-plan>, <complex-act>)
```

Effects of acts are represented using the

`ActEffect( <some-act-i>, <effects-of-act-i>)`

predicate. This predicate specifies mental actions of believing to be performed so as to update the set of beliefs after performing an act. Acts can also have preconditions that are specified using the

`PreconditionAct( <preconditions-of-act-i>, <some-act-i>)`

predicate.

2.2 The acting executive

Requests to perform an action are serviced by the acting executive (see Figure 1). The request (which is represented as an act node) gets scheduled on an acting queue maintained by the executive. This represents the agent's intentions. Plans are structured using *control actions* that when interpreted affect the queue of intentions. Our repertoire of control actions includes sequencing (`snsequence`), conditional (`snif`), iteration (`sniterate`), and a few others (see [2]). *External actions* affect the external world via their respective associated procedures. The acting executive uses SNIP to derive plans, plan decompositions, and the effects and preconditions of actions. It schedules *mental actions* to believe effects of actions. It also schedules acts to achieve preconditions of actions in case they are not satisfied.

---

or-entailment, numerical entailment, and-or, thresh, and non-derivable. The predicate used here represents only the typical antecedent-consequent type of rules. However, the discussion applies to all SNePS rules in general. See [10] for details on the SNePS representation of rules.

[3] As in the case of rules, this representation is being used to facilitate discussion at a general level. Each predicate mentioned here is represented as a structured proposition node. The exact syntax and semantics of these representations can be found in [2, 11].

2.3 Using belief revision

We are using SNeBR as a Truth Maintenance system underlying the knowledge-base. Beliefs currently held by the system are maintained in a belief context. The ATMS ensures consistency of beliefs at all times. Effects of acts lead to adding or deleting of new beliefs. We use the SNeBR operations *add-to-context* and *remove-from-context* to add or remove new beliefs as a consequence of performing some act. The calls to these operations are used to model the two mental actions of *believing* and *disbelieving*.

add-to-context adds a belief to the hypothesis set of the context. Similarly, *remove-from-context* removes a belief from the hypothesis set of the context. SNeBR ensures that all derived beliefs that used the removed hypothesis as an assumption in their derivation are also automatically removed. This is especially useful in the implementation of conditional plans.

2.3.1 Conditional plans

A conditional plan, such as

> "If a block is on a support then a plan to achieve that the support is clear is to pick up the block and then put the block on the table."

is represented in SNePS as a rule approximately like the Predicate Calculus rule

$$\forall x, y[Block(x) \wedge Support(y) \wedge On(x,y) \Rightarrow PlanGoal(Sequence(Pickup(x), Put(x, Table)), Clear(y))].$$

In a situation in which block A is on block B, and the system must clear B, it will derive and store the plan,

$$PlanGoal(Sequence(Pickup(A), Put(A, Table)), Clear(B))],$$

which says that a plan to clear B is to pick up A and put it on the table.

Since this plan is stored, it would seem that it would be retrieved as a plan for clearing B in some later situation when C, for example, is on B, and this would be wrong. However, the plan is derived based on the assumptions $Block(A), Support(B)$, and $On(A, B)$. As soon as A is picked up, the assumption $On(A, B)$ is removed from the current context, and the plan is unavailable until A is put back on top of B. Thus, the representation of conditional plans is correct in systems that include an appropriate belief revision mechanism.

2.4 The natural language component

The natural language understanding component is implemented in a Generalized ATN grammar and is used for analyzing sentences and for generating English responses. The system begins with an empty knowledge-base. In the role of informant, we interact with it using English sentences about the domain, instructing it about the various actions that it can do, and how to solve problems in that domain. The input sentences are analyzed using a domain-specific grammar, the results of which are new beliefs in the knowledge-base. A natural language generation grammar takes the new beliefs and expresses them back in English to show the system's understanding to the informant. Requests to do some action are sent to an acting executive that may then generate and execute a plan to fulfill the request. The informant may also ask questions about plans and the way the system would solve various problems.

During the work described here, it became clear that natural language sentences about planning could be classified into groups, with associated syntactic natural language markers. In the current domain, Blocksworld, we classify the sentences as follows:

1. **Domain description** These types of sentences are simple declarative statements about the state of the domain, *e.g.*, "A is a block", "A is on the table".

2. **Constraint description** These types of sentences are statements about the constraints on objects in the domain, (*e.g.*, "If a block is on a support then the block is not on another support") independent of a the agents capabilities.

3. **Act/Plan description** These types of sentences define the types of actions the modeled agent is capable of performing, their associated preconditions and effects as well as how they are composed to build complex plans.

4. **Performative requests** These types of sentences are simple imperative requests to perform an action, *e.g.*, "Pick up A".

Type (1) sentences generaly are simple copular clauses, or 'There-is' introductory sentences (*e.g.*, "There is a table"). Noun phrases are generally definite, with proper name reference being the rule, rather than the exception. Type (4) sentences are simple imperative requests whose object noun phrases may be definite or indefinite, (*e.g.*, "Pile A on B on C", "Put A on a block").

Sentences of type (2) and (3) are rule-like (similar to FOPL rules), in their structure and use of variables. In our domain, constraint description sentences map directly to FOPL rules; *e.g.*, "If a block is on a support then the block is not on another support" becomes[4]

$$\forall x, y, z[(Block(x) \wedge Block(y) \wedge Block(z) \wedge On(x, y)) \Rightarrow \neg On(x, z)].$$

Act/Plan description sentences have less direct mapping, but in both the treatment of definite/indefinite noun phrases is the same. In the context of a sentence of these types, an indefinite noun phrase introduces a new variable of the type associated with the common noun of the noun phrase. Definite noun phrases refer back to explicitly introduced (via an indefinite noun phrase) variables in the same sentence. The syntax of these types of sentences is well marked (by words like "before", "after", "then") and is highly decomposable into simpler clauses of types (1) and (4). For instance, a plan to pile three blocks consists of three repeated type (4) sentences (*e.g.*, "put the first block on the table", "put the second block on the first block", etc.) expressed in the appropriate order with explicit sequence separators ("... and then ..."). Similarly, preconditions and effects have antecedent and consequent clauses of type (1). In both type (2) and (3) sentences, the noun phrase positions associated with component clauses are generally variables, although definite reference to objects in the domain ("the table") also occur.

In this domain, sentences about plans display a useful compositionality, in that more complex types of sentences (describing more complex types of knowledge) can be built from the simple clauses associated with describing a domain and performing acts; where they differed was in the treatment of definite/indefinite noun phrases as referring to and introducing variables, respectively.

3 Discussing, using, and recognizing plans

The system is capable of discussing its beliefs. For example, consider the the queries

Is A on B?
`Yes, A is on B.`

Is A on C?
`I really don't know if A is on C.`

The answer to the second query above is inconclusive because there is no way for the system to confirm or deny the queried fact. To be able to do so, we can instruct the system about domain-specific rules in English. For example, to answer the above query conclusively, it needs the following rule:

[4] The conjunct $x \neq y$ is not needed in the antecedent of this rule because SNIP uses the Unique Variable Binding Rule (UVBR) [9] which prevents x and y from binding to the same term.

If a block is on a support then the block is not on another support.
I understand that if a block is on a support
then the block is not on another support.

Is A on C?
No, A is not on C.

3.1 Discussing plans

Similarly, we can ask the system to answer questions involving plans. SNIP, the plan decision procedure, is used to derive an appropriate plan and respond to the query. For example,

How would you pile A on C on B?
I understand that a plan for performing pile on A and C and B is
by performing put on B and a table and then performing put on C and B
and then performing put on A and C.

Notice that in this case, a plan is derived but not executed. So far we have demonstrated that the system is able to interact with the user about its beliefs, and it can use the domain rules to answer queries about the domain. The system can also understand natural language domain descriptions of domain-specific rules and plans and acts as contained in the following paragraphs

> *There is a table. The table is a support. Blocks are supports. Picking up is a primitive action. Before picking up a block the block must be clear. After picking up a block the block is not clear and the block is held. If a block is on a support then after picking up the block the block is not on the support and the support is clear. Putting is a primitive action. Before putting a block on a support the block must be held and the support must be clear. After putting a block on a support the block is not held and the block is clear and the block is on the support. After putting a block on another block the latter block is not clear.*
>
> *A plan to achieve that a block is held is to pick up the block. A plan to achieve that a block is on a support is to put the block on the support. If a block is on a support then a plan to achieve that the support is clear is to pick up the block and then put the block on the table. A plan to pile a block on another block on a third block is to put the third block on the table and then put the second block on the third block and then put the first block on the second block.*

3.2 Using plans

Given the above description of a domain, and a situation like

A is a block. B is a block. C is a block. C is on A. C is clear. B is clear and on the table.

the system can use the domain description from above to plan and act in the situation. Thus, it will be able to derive plans and perform actions required to fulfill a request like

Pile A on B on C.

3.3 Recognizing plans

Our representations for plans and acts also facilitate plan recognition. We have implemented a system which allows the deduction of a set of plans some agent might be performing from information about the acts that agent has been performing. This plan recognition system has mainly been applied to a simple version of the Blocksworld, but was also used for a tutoring domain in order to demonstrate

how domain knowledge can be used to narrow the number of possible plans some student might be engaged in (see [12]).

In order to explore the advantages and disadvantages of using node-based inference and path-based inference, corresponding rules were implemented and tested in the Blocksworld. The plan recognition system which uses node-based inference was able to deal with complex acts and subplans, *i.e.*, it could identify a plan even if the reported acts are only implicitly represented in complex acts for which there are separate plan-act rules which contain the reported acts explicitly. The problem with node-based inference for plan recognition is that it generates a lot of nodes in order to make the component relations explicit, and it cannot deal with uninstantiated plan-rules due to the current implementation of quantified variables.

Using nested entailments, a left-recursive representation for plans, and a special representation for the result of a plan recognition process, it was possible to use mainly path-based inference, which increases efficiency and avoids some of the problems related to the quantification of variables in plan-rules. The plan recognition rules were tested successfully for instantiated and uninstantiated plan-rules and for plan-rules with complex acts and corresponding subplans.

Assuming the plan recognition system described in this paper is part of the modeled agent's mind, the plan recognition process can be described as follows: the modeled agent is told that a third agent is performing certain acts. Using its knowledge base of plan-act and plan-goal propositions, the modeled agent identifies those plans which contain the reported acts in the correct temporal order and concludes that the agent might perform the corresponding acts or might try to achieve the corresponding goals.

4 Future research issues

Our goal is to model a rational cognitive agent whose behavior is driven by its beliefs, desires, and intentions. We want our agent to understand natural language, reason about beliefs, act rationally based on its beliefs, recognize plans, and do plan-based text generation. Doing all these tasks in a single coherent framework poses several constraints. We are discovering that SNePS and its underlying theories contribute effectively towards our goal. We have designed and implemented intensional propositional representations for plans. This is a major advance over operator-based descriptions of plans. Operator-based formulations of actions tend to alienate the discussion of operators themselves. Operators are usually specified in a different language than that used for representing beliefs about states. Moreover, plans (or procedural networks) constructed from these operators can only be accessed by specialized programs (critics, executors) and, like operators, are represented in still another formalism. Our representations for acts, actions, goals, and plans build upon and add to the intensional propositional representations of SNePS. This framework enables us to tackle various tasks mentioned above in a uniform and coherent fashion.

Our current system is being advanced in several directions. In the context of planning, there are issues associated with conjunctive goals [14], non-linear plans [8, 13, 1], and dealing with the effects of actions.

Language used in planning contexts is slightly more constrained than that in arbitrary discourse. Sentences describing plans tend to be declarative, with a syntactically decomposable structure involving goal, effect, and plan definition. Handling reference is simplified by the assumption that common noun phrases correspond to typed variables. Indefinite noun phrases introduce new variables, definite noun phrases refer to previously introduced variables. Natural language generation of plans and rules involves careful selection of relevant attributes of these variables.

4.1 Sensory acts and external events

So far, we have concentrated on the problem of designing representations suitable for discussing, using, and recognizing plans. We have demonstrated their use in a single-agent world. We are now ready to explore issues involved in using our representations to model rational cognitive agents

that are capable of acting in the real world. The real world is constantly undergoing change in the presence of several agents (including the modeled one) as well as by natural phenomena. In order to behave as rational agents in a real world, they should be endowed with appropriate sensors as well as effectors. Thus, we are ready to explore issues concerning sensory acts, external events, and how an agent's beliefs, desires, intentions, and actions are affected by them.

4.2 Integrating inference and acting

In our current model (and in other state-of-the-art systems), reasoning is preformed by some inference engine and acting is done under the control of an acting executive. In order to achieve our goals, we have come to the conclusion that inference and acting need to be more tightly coupled. A survey of most systems will reveal that it is somewhat awkward to do acting in reasoning (or logic-based) systems (but it is convenient to talk about representational and reasoning issues using them), and it is awkward to study reasoning and representational issues in systems designed for acting/planning. We are beginning to take the viewpoint that logical reasoning rules implicitly specify the act of believing, and the process of reasoning can be treated as specialized (more efficient) acting. This will enable us to integrate the acting and inference engines that can be driven by regular reasoning rules as well as connectives that will transduce a belief status to an intention-to-act status. We are currently designing such connectives. Thus, our future research will attempt to clarify the relationship between inference and acting. This integrated approach used in conjunction with the principles underlying propositional semantic networks will preserve the power of acting, as well as reasoning systems, and provide a richer framework within which one can experiment with various modeling issues in AI. Some preliminary results can be found in [3].

4.3 Structured variables

Another direction for our future research involves a reexamination of the representation of variables in SNePS. Consider again the conditional plan,

$$\forall x, y[Block(x) \wedge Support(y) \wedge On(x,y) \Rightarrow \\ PlanGoal(Sequence(Pickup(x), Put(x, Table)), Clear(y))].$$

As in the FOPC representation of this rule, the SNePS representation contains the subexpression $Pickup(x)$. Although in SNePS, the variable x is connected in the network to its restriction, $Block(x)$, the variable x is still an "atomic" node, and the term $Pickup(x)$ does not contain the restriction on x as a subterm of it. The significance of this is that the act "pick up a block" is not represented by a single term in the plan expression. Compare this representation to something like:

PlanGoal(Sequence(Pickup(x:Block s.t. On(x, y:Support)),
Put(x:Block s.t. On(x, y:Support), Table)),
Clear(y:Support)).

Here, each sub-expression is conceptually complete. For example, *Pickup(x:Block s.t. On(x, y:Support))* clearly represents the act of picking up a block that is on a support. (It should be noted that the SNePS representation, using a network syntax, would not be as redundant as the linear representation.) We plan to investigate these representational issues further.

4.4 Applications to simulate agents

A more application-oriented direction we may pursue is to apply our techniques of representing and reasoning about plans to simulate some human agent, and to try to predict what that human agent would do in certain hypothetical circumstances.

4.5 Increased plan recognition

In order to recognize conditional or iterative plans, the relevant conditions which are true at the time the reported acts were performed must also be reported in addition to the reported acts. The deduction rules for plan recognition would have to be extended to include the control actions `snif` and `sniterate`.

Since the acting executive automatically schedules acts to achieve preconditions of other scheduled acts, presumably other agents do the same thing. A plan recognizer should use the precondition rules to account for those acts which are not explicit in the plan-act or plan-goal rules.

The belief revision system SNeBR can be used to implement incremental plan recognition. The belief revision system can be used to discriminate between potential plans or goals by establishing a separate context for each plan (or combination of plans) and keeping track of the consistency of each context as more and more acts about the agent are reported. The number of assumptions supporting a conclusion that the agent is performing a particular plan or pursuing a particular goal indicates the probability of that conclusion and thus can also be used to discriminate among competing plans or goals.

5 Summary

In this paper, we have given an overview of the architecture of the SNePS acting system. The current version of the SNePS acting system can understand natural language utterances about a blocksworld. We reviewed our representations of plans and acts, the design of the acting executive, how we make use of a belief revision system, and the design of the natural language component. We took a brief look at how the system is able to discuss, use, and recognize plans in a Blocksworld domain. Lastly, we presented several future research issues related to this work that we are currently working on.

References

[1] Mark Drummond and Austin Tate. AI planning: A tutorial and review. Technical Report AIAI-TR-30, Artificial Intelligence Applications Institute, University of Edinburgh, Edinburgh, November 1987.

[2] D. Kumar, S. Ali, and S. C. Shapiro. Discussing, using, and recognizing plans in SNePS preliminary report — SNACTor: An acting system. In *Proceedings of the Seventh Biennial Convention of South East Asia Regional Confederation*, pages 177–182, New Delhi, India, 1988. Tata McGraw-Hill.

[3] Deepak Kumar. An integrated model of acting and inference. In Deepak Kumar, editor, *Proceedings of the First Annual SNePS Workshop*, November 1989.

[4] Deepak Kumar, Syed S. Ali, and Stuart C. Shapiro. Discussing, Using, and Recognizing Plans in SNePS–Preliminary Report. In Scott S. Campbell and Paul W. Palumbo, editors, *Proceedings of the Third Annual University at Buffalo Graduate Conference in Computer Science*, pages 62–69, SUNY at Buffalo, Buffalo, NY, 1988. Department Of Computer Science.

[5] J. P. Martins and S. C. Shapiro. Belief revision in SNePS. In *Proceedings of the Sixth Canadian Conference on Artificial Intelligence*, pages 230–234. Presses de l'Université du Québec, 1986.

[6] J. P. Martins and S. C. Shapiro. Theoretical foundations for belief revision. In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning About Knowledge*, pages 383–398. Morgan Kaufmann Publishers, Los Altos, CA, 1986.

[7] J. P. Martins and S. C. Shapiro. A model for belief revision. *Artificial Intelligence*, 35(1):25–79, 1988.

[8] Earl D. Sacerdoti. *A Structure for Plans and Behavior.* Elsevier North Holland, New York, NY, 1977.

[9] S. C. Shapiro. Symmetric relations, intensional individuals, and variable binding. *Proceedings of the IEEE*, 74(10):1354–1363, 1986.

[10] S. C. Shapiro and The SNePS Implementation Group. *SNePS-2 User's Manual.* Department of Computer Science, SUNY at Buffalo, 1989.

[11] Stuart C. Shapiro, Beverly Woolf, Deepak Kumar, Syed S. Ali, Penelope Sibun, David Forster, and Scott Anderson. Discussing, using, and recognizing plans–Annual Report for 1988. Technical report, North-East Artificial Intelligence Consortium, 1989.

[12] Stuart C. Shapiro, Beverly Woolf, Deepak Kumar, Syed S. Ali, Penelope Sibun, David Forster, Scott Anderson, James Pustejovesky, and Juergen Haas. Discussing, using, and recognizing plans–Project Report. Technical report, North-East Artificial Intelligence Consortium, 1989.

[13] A. Tate. Generating project networks. In *Proceedings 5th IJCAI*, pages 888–93, 1977.

[14] R. Waldinger. *Achieving Several Goals Simultaneously*, pages 94–136. Ellis Horwood, Chichester, England, 1977.