In Proceedings of the National Conference of the American Association for Artificial Intelligence, pages 597-604, Orlando, FL. August, 1999.

Cooperative Plan Identification: Constructing Concise and Effective Plan Descriptions

R. Michael Young
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7534
young@csc.ncsu.edu

Abstract

Intelligent agents are often called upon to form plans that direct their own or other agents' activities. For these systems, the ability to describe plans to people in natural ways is an essential aspect of their interface. In this paper, we present the Cooperative Plan Identification (CPI) architecture, a computational model that generates concise, effective textual descriptions of plan data structures. The model incorporates previous theoretical work on the comprehension of plan descriptions, using a generate-and-test approach to perform efficient search through the space of candidate descriptions.

We describe an empirical evaluation of the CPI architecture in which subjects following instructions produced by the CPI architecture performed their tasks with fewer execution errors and achieved a higher percentage of their tasks' goals than did subjects following instructions produced by alternative methods.

Introduction

Complex activities, by definition, contain a large amount of detail. When people describe activities to one another they leave out information they feel is unimportant and emphasize information they feel is essential. This economy of communication is an example of speakers obeying Grice's maxim of Quantity: say no more and no less than what is needed (Grice 1975). There is a wide range of contexts where intelligent agents that create and use plans might require the ability to generate task descriptions of similar brevity. Unfortunately, it is not a straightforward matter to produce an effective description of a given plan automatically when one or more of the intended readers or hearers are human. There is a mismatch between the amount of detail in a plan for even a simple task and the amount of detail in typical plan descriptions used and understood by people.

In this paper, we consider communication in the context called *plan identification*. In this context, a speaker describes a plan P to a hearer in order to single out P as the solution to what the speaker believes is a mutually understood planning problem. A *description* of a plan P is a subset of the components of P used by a speaker for plan identification.

In the discussion that follows, we define a computational model of plan identification, called *cooperative plan identification* (CPI), used to generate concise descriptions of plans produced by AI planning systems. We also describe a task efficacy evaluation (Walker & Moore 1997) of the cooperative plan identification techniques where plan descriptions are used as instructions for tasks carried out by human subjects. Subjects carry out their tasks in a simulated domain and their performance on the tasks is measured to determine the effectiveness of the instructions that they follow. The experiment demonstrates that the descriptions produced by cooperative approaches are more effective than those produced by several alternative techniques.

Related Work

While several natural language systems have been developed for the generation of textual descriptions of action, these systems have been limited in the effectiveness of the descriptions they produce by the complexity of the activities that they describe. Mellish and Evans (1989) describe a system that produces textual descriptions of plans created by the NONLIN planner (Tate 1977). Their system generates text that contains reference to every component in a NONLIN plan. Consequently, as Mellish and Evans themselves point out, the resulting descriptions often contain an inappropriately large amount of detail.

Vander Linden and Martin (1995) discuss a text generation system that produces texts describing small sets of plan components, focusing on the selection of rhetorical relations that best expresses the components' procedural relationships with other actions in the same plan. The Drafter project at the University of Brighton (Hartley & Paris 1997) has developed a system to exploit a plan-based representation of activities to support multilingual instruction generation. This system represents task domains in a common action language and generates instructions based on the plans for a given task. In both these systems, plans for specific tasks are constructed by hand; because the detail present in the plans is pre-determined by human users, the systems

do not need to address issues of plan size and can avoid the complexity introduced by dealing with larger, automatically generated plans.

A Cooperative Approach to Plan Description

In this research, we adopt the view that the use of plan descriptions in discourse is an instance of Gricean cooperation. A central idea in this work is that Grice's maxim of Quantity guides a speaker when he is selecting the amount and type of detail to include in a plan description. Under this interpretation, a candidate plan description contains sufficient detail precisely when a hearer can reconstruct the plan being described (or one reasonably close to it) from the content present in the candidate.

To produce cooperative plan descriptions, we use a generate-and-test architecture called *cooperative plan identification* (CPI). The process used by the architecture is divided into two functions (the overall architecture is shown in Figure 1). The first, called the *generator* function, constructs candidate descriptions; the second function, called the *evaluator*, tests descriptions against success criteria captured by the interpretation of the maxim of Quantity and described in the following section. The algorithm searches the space of descriptions of the plan that the speaker is identifying (called here the *source plan*), looking for an acceptable plan description of minimal size and structure. Searching the space of all plan descriptions is computationally expensive; rather than perform an exhaustive search for candidate descriptions, we describe two algorithms used as CPI generator functions that restrict the space they search to more tractable subsets of the full space while still producing reasonable candidates.

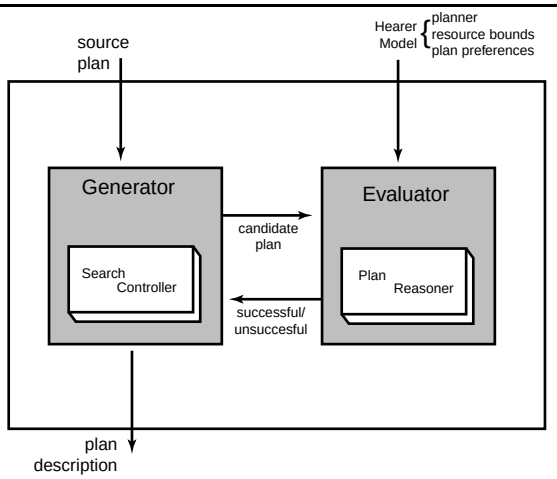

Figure 1: An Overview of the CPI Architecture.

Cooperation in Plan Description

Our previous work (Young 1996) describes a technique for characterizing the adequacy of a plan's description with repect to the amount of detail it contains. We adopt this technique as the CPI evaluator function and describe it briefly in this section. See (Young 1996) for a more complete description.

In our previous work, the plan description process is modeled as a collaboration between a speaker and a hearer. In order to understand a plan description, a hearer uses her knowledge about plans and planning to fill in any information that was missing from a speaker's description. To produce a plan description that is cooperative, a speaker uses his knowledge about the hearer's interpretation process to select a plan description that is brief but contains enough information to be understood. In this work, the hearer's interpretation process is represented by the use of a plan reasoning algorithm that takes a partial plan description and fills in its gaps, performing the same type of plan reasoning that a planning system would use to create the source plan in the first place. If the resulting complete plan (or plans) is similar enough in structure to the source plan, the algorithm characterizes the candidate as acceptable.

In this work, the interpretation of a candidate plan description is represented as search through a space of partial plans represented as a graph. The construction of the graph is controlled by a *plan-space* planning algorithm (Kambhampati, Knoblock, & Qiang 1995) that models the plan reasoning the hearer employs to reconstruct any detail missing from the candidate description. In addition to the plan reasoning algorithm, there are two other components that are central to the hearer model we use. The first is a limit on the hearer's reasoning resources — specifically, the amount of plan reasoning that she can bring to bear during the interpretation process. Clearly, the effort needed to construct a complete plan from a partial one requires the use of a hearer's reasoning resources. These resources are finite; a cooperative speaker that takes the hearer's use of these resources into account can adjust the content of his description so that the resources will not be exhausted. Here this resource limit is represented by an integer constant that places an upper bound on the number of nodes that can be searched in the plan-space graph when characterizing a candidate plan description.

The second central issue in the representation of the hearer's plan reasoning is her use of plan preferences, that is, her preferences for plans of particular structure over others. As a hearer fills in the gaps in a partial plan description, her planning activity is influenced by her preferences over aspects of the task domain. Her preferences for types of actions, for particular sequences of actions to achieve a goal, etc, all influence the structure of the complete plan that will emerge. Plan preferences are represented here by a ranking function that assigns non-negative integer rankings to plans on the fringe of the search space during construction of the graph. Regions of the graph rooted at nodes ranked most pre-

ferred will be explored before regions rooted at nodes with less preferred rankings.

The three aspects of the hearer's plan reasoning model (i.e., her planning algorithm, her resource limits and her plan preferences) operate together to construct a plan space graph representing the inferences that will be performed by the hearer when interpreting the candidate plan description. The structure of this graph determines the adequacy of the candidate to serve as the source plan's description: when all of the solution plans in the graph are reasonably similar to the source plan, then the candidate description is considered acceptable. An optimum plan description is an acceptable candidate description containing the fewest number of plan components of all similarly qualified descriptions.

While this work provided an algorithm for determining the adequacy of a given candidate description, we did not describe an algorithm for constructing candidates in an efficient manner. In the following section, we describe an architecture that combines our previously defined adequacy criteria for candidate descriptions with several related algorithms for efficiently generating candidates.

Generating Candidate Plan Descriptions

Generating an effective plan description is a difficult problem since there is no obvious technique for constructing the description directly from the source plan. While exhaustive search through the space of all the possible candidates is guaranteed to find an optimum description, the computational cost of this search is prohibitive; since every coherent subset of the source plan's components could potentially serve as the plan's description, the space of candidate descriptions is essentially the power set of the set of components in the source plan.

There is evidence, however, that optimal plan descriptions are not required for natural, concise and effective descriptions. As reported by Hull and Wright (1990), people often generate non-optimal plan descriptions and when they do so, their readers or hearers are still able to carry out the tasks at hand. A problem closely related to plan identification, the task of generating referring expressions, is similarly constrained by efficiency limitations when algorithms search for optimal descriptions. As Dale and Reiter (1995) describe in their characterization of computational models used to generate referring expressions, they adopt the strategy of restricting search to tractable subsets of the solution space in such a way that the systems generate concise (but not necessarily optimal) texts that are both natural and effective. This strategy has been adapted for use in plan identification and is discussed further below.

This section defines four implemented algorithms used to determine a set of plan components that will serve as a source plan's description. The first two are *cooperative techniques*, motivated by distinct computational interpretations of Grice's maxim of Quantity. These two algorithms serve as generator functions in implementations of the cooperative plan identification architecture. The second pair of algorithms represent approaches that do not take a model of the hearer into account, using instead two distinct techniques that directly translate the source plan into its description. These two *direct translation techniques* are used in the evaluation described below in order to provide a basis for comparison against the two cooperative techniques. Space limitations preclude a detailed comparison of example data structures created by these algorithms; sample data structures and the texts that correspond to them are described in (Young 1997).

In this paper, we will use the DPOCL planner (Young, Pollack, & Moore 1994) as the hearer model's planning algorithm. DPOCL extends the UCPOP planner (Penberthy & Weld 1991) by incorporating hierarchical planning directly into a causal link framework. DPOCL's principal qualification for use in this work is that it is not built especially for the generation of task descriptions; rather, it is a domain-independent planning algorithm. DPOCL plans contain sufficient structure to ensure the plans' soundness and, consequently, the plans serve as strong test cases for the generation of plan descriptions. In addition, DPOCL is readily characterized as a plan-space planning algorithm.

Local Brevity: Exploiting a Plan's Structural Information. The *Local Brevity* algorithm searches for acceptable plan descriptions moving through the space of candidate descriptions from complete, detailed candidates toward partial, abstract ones. In this manner, the algorithm is similar to the Local Brevity algorithm of Dale and Reiter (1995); as in their approach, the CPI Local Brevity generator begins its search with a complete description (a complete plan) and creates new candidates by iteratively removing single components from the description based on local decisions dictated by a set of heuristics. These heuristics are based on results from studies of the comprehension of instructional and narrative texts (described below) that indicate that the presence of some plan components in a plan's description are more important to the hearer's understanding of the plan than are others. To determine the order in which components are deleted from the working description, the heuristics are used to assign a weight to each of the source plan's components. The algorithm iterates, first deleting the element in the plan that is weighted lowest, then passing the resulting working description to the evaluator function. Because the Local Brevity algorithm begins its search with a complete plan, the initial description is likely to be acceptable to the evaluator (that is, it is likely to contain sufficient information to properly identify the plan being described). The deletion process iterates until the evaluator indicates that the working plan has become too partial to be acceptable. At this point, the algorithm adds back in the last component that was deleted and

uses the resulting data structure as the source plan's description.

The heuristics used to determine the order in which plan components are deleted are captured in two weighting functions, one used to rank plan steps and one used to rank causal links. These weighting functions each sum a number of terms representing the contribution of structural features of the plan to the importance of the component's appearance in the plan's description.

A plan's steps are weighted based on three factors reflected in the three terms in Equation 1 below. The equation is motivated by the following heuristics suggested by the more qualitative results described in (Trabasso & Sperry 1985; van den Broeck 1988; Graesser *et al.* 1980):

- The greater the number of causal dependencies a step has on previous steps in the plan, the more important the appearance of the step is in the plan's description.
- The greater the number of subsequent steps that depend upon a step, the more important the appearance of the step is in the plan's description.
- The deeper a step appears in the plan hierarchy, the less important the appearance of the step is in the plan's description.

For a given step s in plan P, the weight w_s assigned to s is determined by the summation of three terms:

$$w_s = (\text{In}(s,P)\times k_p)+(\text{Out}(s,P)\times k_e)+(\text{Depth}(s,P)\times k_d) \quad (1)$$

In this equation, $\text{In}(s,P)$ is a function returning the number of s's satisfied preconditions in P (i.e., the number of causal links leading in to s), k_p is a constant scaling factor for incoming causal links, $\text{Out}(s)$ is a function returning the number of causal links leading out of s, k_e is a constant scaling factor for outgoing causal links, $\text{Depth}(s)$ is a function returning the number of ancestors of s in P, and k_d is a constant scaling factor for step depth.

The values of the scaling factors are determined empirically and may vary between domains. All scaling factors in Equation 1 except k_d are constrained to be no less than 0 while the magnitude of k_d is constrained to be no greater than 0.

A single factor is used to assign weights to the causal links in a plan description: links are weighted based on their temporal duration. Results from reading comprehension and text summarization studies (Golding, Graesser, & Millis 1990; Kintsch & Van Dijk 1978; Rumelhart 1977) suggests that causal relationships between steps that are temporally close are often so readily reconstructed that references to the relationships are elided from plan descriptions. In causal link planners without an explicit representation of time (such as DPOCL), an estimate of the link's duration can be made by counting the number of steps in the plan that might possibly occur between the link's source step and its destination step. The greater the number of intervening steps, the longer the duration of the causal link. In the CPI implementation, for a given causal link l from step s_i to step s_j in plan P, the weight assigned to l is expressed by the equation

$$w_l = (\text{Inter(l, P)} \times k_l) \quad (2)$$

where $\text{Inter}(l,P)$ is the number of all steps that could possibly intervene between s_i and s_j in P and k_l is a constant scaling factor for intervening steps. k_l is restricted to be no less than 0.

To determine the sequence of components to be eliminated from the source plan, both the components' weights and their position in the plan structure are considered. In general, components with lower weights are eliminated first. However, in order to preserve the decompositional structure of the partial plan (in accordance with the constraint on referential coherence described above), steps are only eliminated from the leaves of the plan (that is, steps are only eliminated when they are either primitive steps or abstract steps whose children steps have already been eliminated). The elimination of causal links is not similarly constrained.

As steps and causal links are removed from the plan, all plan components that make reference to those steps and links are also eliminated. For instance, when a step is removed from a plan description, all causal links leading into or out of that step are removed, all ordering constraints for the step are taken out of the plan description and the step's binding constraints are also deleted.

The Plan Path Algorithm: Following the Source Planner. The *Plan Path* algorithm generates candidate descriptions following a path through the space of plans created by the source planning system as it solved the original planning problem. The Plan Path algorithm begins its search by considering the null plan at the root of this graph and moves through the graph by selecting at each choice point the child node that lies along the shortest path from the root node to the source plan. When the algorithm visits a node in this space, it sends the partial plan associated with that node to the evaluator, testing to see if the plan can serve as a description. The Plan Path algorithm halts as soon as it finds a plan that is successful (that is, a plan that contains enough information to effectively identify the plan being described).

This algorithm requires access to the list of nodes that lie along the path from root to source plan in the source planner's plan graph. In the CPI implementation, the nodes are supplied as input by the source system along with the source plan. Providing these additional data structures is a minor requirement for a refinement planning system, since the nodes are created by the source planner during the search that produces the source plan to begin with.

Two Direct Translation Algorithms. In order to provide comparisons to the cooperative plan identification algorithm, two direct translation approaches are also defined. The implementation of these approaches is described briefly below.

The Exhaustive Algorithm: The component of Mellish and Evans's system that generated the content of a plan description was relatively straightforward: the system generated a description that referred to every component of the plan being described (with the exception of certain NONLIN bookkeeping structures). The same strategy for content selection was used here in the *Exhaustive* algorithm. To generate plan descriptions, the Exhaustive algorithm takes as input the source plan and, since every element of the plan is to be included in the description, the process returns the complete source plan as its output.

The Primitive Algorithm: One possible approach to describing a plan is to describe just the lowest-level steps in the plan – those that will actually be executed. These steps correspond to the primitive steps in a DPOCL plan, the leaf node steps in the source plan data structure. The *Primitive* algorithm takes as input the source plan and selects as the plan description the primitive steps in the plan, returning those steps in a total temporal order consistent with the source plan's temporal constraints.

Empirical Evaluation

To evaluate the CPI model, we studied the empirical validity of the claim that providing conversational participants with a cooperative description of a plan increases the effectiveness of the communication. To address this claim, human subjects were presented with a series of text descriptions whose content had been automatically generated by the four algorithms described above. The subjects were asked to carry out the plan descriptions in a simulated task domain. Their actions were then analyzed along several dimensions to determine the quality of the subjects' performance. The hypothesis for the experiment stated that subjects that followed instructions produced by the cooperative techniques (i.e, the Local Brevity and Plan Path algorithms) would perform their tasks with fewer errors and achieve more of their top-level goals than subjects following instructions produced by the alternative approaches (i.e., the Exhaustive and Primitive algorithms).

The process used to produce the texts in this experiment was divided into three main components. The first module, consisting of the DPOCL planning algorithm, was used to construct solution plans for four planning problems in the task domain. The plans produced by DPOCL were then passed to the second component, a content determination module. For each input plan, this module applied each of the four approaches to content determination discussed above. The two generate-and-test approaches and the two alternative direct-translation algorithms each generated a corresponding plan description, resulting in a total of four descriptions for each input plan. Finally, the four plan descriptions were passed to a text realization module. The realization module determined the English text used to describe the plan components included in the descriptions as well as the order that the text appeared in the text descriptions.

During the experiment, 24 human subjects [2] were individually asked to perform a series of four tasks — one for each of the four experimental source plans.[3] For each task, we provided each subject with a list of the goals for each task (taken from the goal specification of the corresponding source plan) and a set of written instructions (one of the four text descriptions that had been produced for the source plan). They were asked to carry out the task as described by the text within a computer simulation constructed using a text-based virtual reality system. The task domain simulated a college campus and subjects' tasks involved running errands across campus (e.g., checking out books from the library, registering for classes at the Registrar's Office). Subjects interacted with the simulation via a command-line interface; the simulation was designed with a one-to-one correspondence between simulation commands and the primitive actions in the operator set used by the planner when creating the source plans for the experiment.

Configuring the Experimental System

The system components described in the preceding section contain a number of user-specifiable parameters. The various settings for the parameters that were used in the experimental systems are described here.

Local Brevity Weighting Functions. The Local Brevity algorithm uses weighting functions to assign weights to each step and causal link in a plan; the weighting functions appear in Equations 1 and 2. The values of the scaling factors that were used in the experiment are as follows: for incoming causal links, $k_p = 1$, for outgoing causal links $k_e = 5$, for step depth, $k_d = 1$, for intervening steps, $k_l = 2$. These values are assigned to reflect my estimation of this relative emphasis of the factors discussed in the research by Trabasso and Sperry and by Graesser *et al* discussed above. [4]

[2] Subjects were solicited from the general University of Pittsburgh community and paid $9.00 per hour for their participation.

[3] Subjects were divided into four groups; each group performed the same set of tasks but was presented with the tasks in an order differeing from the order used to present the tasks to the other groups.

[4] These constants are user-specifiable parameters and, short of performing extensive experiments that compare the performance of the system under various settings, no strong conclusions can be drawn about the *relative* merits of one set of values over any others.

The Hearer Model. The CPI hearer model contains three customizable parameters: the planning algorithm, the hearer's plan preferences and her plan reasoning resource limit. DPOCL, the planning algorithm used in the experiment as the model of the hearer's plan reasoning, is described in detail in (Young, Pollack, & Moore 1994). The plan ranking function used in the experiment employed a domain-independent metric, looking only at the size of the plan, preferring short, hierarchically structured plans with few top-level steps. In the absence of empirical evidence to suggest specific values for hearers' plan reasoning resource bounds, an objective method was devised to automatically generate a setting for the limit used for each plan being described. Using this method, the mid-point is found between the greatest depth bound where a complete plan description is generated and the least depth bound where an empty plan description is generated. To compute the mid-point value, each algorithm's depth bound is initially set to 0. A plan description is generated using this depth bound setting, and the depth bound value is incremented until a plan description is generated that contains less than the complete structure of the source plan. This value is taken as the lower bound of the range for the resource bound. The process continues to iterate, incrementing the depth bound and producing a new description, until the description that is produced contains no detail at all. This value is taken as the upper bound of the range for the resource bound. The mid-point between the upper and lower bounds is then used as the depth bound when generating a description for that source plan. Although the use of this technique results in the assignment of depth bounds that vary between plans, the method provides a basis for comparison by defining the same relative point in the space of all candidates that each algorithm considers.

Summary of Results

The data that was collected for each subject consisted of a series of four *executions*. Each execution represents the sequence of all commands typed by the subject. Because of the one-to-one correspondence between elements of a command (i.e., command names, argument names) and the act-types, locations and objects in the simulation domain, it was straightforward to translate each subject's exections into a sequence of fully-instantiated primitive plan steps in the language of the planner.

To measure the success of the subject's execution, we used three dependent variables:

The Step Failure Ratio (SFAIL). The percentage of the total number of steps containing preconditions that failed during the execution.

The Precondition Failure Ratio (PFAIL). The mean percentage of the number of preconditions for a failed step that were unmet when the step was executed.

The Goal Failure Ratio (GFAIL). The percentage of the plan's top-level goals that were unachieved when the execution ended.

For each dependent variable, data was averaged over items (that is, over plans) and a two-way repeated-measures ANOVA was conducted. Means and standard deviations for these variables, along with the results of the various analyses of variance, are shown in Table 1. In order to determine if the experimental source plans themselves had an effect on subjects' performance, a separate analysis was performed for each item, using a one-way, between-subjects ANOVA. Results of the second analysis are described in depth in (Young 1997) and are mentioned briefly below.

The patterns of means for each of the dependent variables clearly support the hypothesis that cooperative plan identification techniques (using the Local Brevity and Plan Path algorithms) produce more effective plan descriptions than the two alternative approaches (the Exhaustive and Primitive algorithms). In particular, the data indicate that the cooperative model has a significant effect on the number of execution errors performed by subjects during tasks ($F(3,63) = 7.06$, $p < .05$) as well as the number of goals left unachieved by subjects during tasks ($F(3,63) = 3.52$, $p < .05$); the effect on PFAIL did not reach statistical significance ($F(3,63) = 2.60$, $p < .08$). See Table 1 for relevant data.

Planned comparisons testing the specific prediction that the cooperative techniques produce more effective descriptions than the direct-translation techniques (Local Brevity and Plan Path *vs.* Exhaustive and Primitive) confirmed this hypothesis for both SFAIL and GFAIL. The comparisons also showed that the Local Brevity and Plan Path algorithms did not differ significantly from each other on any of the three dependent variables. See Table 2 for all relevant F-values. This table indicates the pairwise relationships between the means for each of the conditions of the experiment.

In order to determine if the differences in the amount of detail contained in the the plan descriptions could have accounted for these results, we performed the two-way analysis of variance for each of the dependent variables a second time, adjusting the number of errors for each subject by dividing the data by the number of components in the corresponding text description. Details of this analysis are found in (Young 1997). The results were the same as those reported for the initial analysis, with the following exceptions. The pairwise comparison between Exhaustive and Plan Path algorithms indicates that the difference between the two on the measure PFAIL and SFAIL were no longer significant ($F(3,63) = 2.44$ and $F(3,63) = 2.57$, respectively).

Discussion

The data clearly show that when subjects follow instructions produced by the cooperative techniques, they make fewer execution errors and achieve more of their

Table 1: Data for the Step Failure Ratio (SFAIL), Precondition Failure Ratio (PFAIL) and Goal Failure Ratio (GFAIL).

Exhaustive		Primitive	
M	*SD*	*M*	*SD*
12.8798	12.8980	7.8929	11.1346

Local Brevity		Plan Path	
M	*SD*	*M*	*SD*
2.3228	6.5711	2.4725	8.5909

Means and Standard Deviations for SFAIL

SOURCE	*df*	SS	MS	F
Algorithm	3	1846.8581	615.6194	7.06
Algorithm × Subject Grp	9	999.1757	111.0195	1.27
Error(Algorithm)	63	5751.9290	87.1504	

ANOVA Summary Table for SFAIL

Exhaustive		Primitive	
M	*SD*	*M*	*SD*
7.8495	7.55756	4.6242	6.92501

Local Brevity		Plan Path	
M	*SD*	*M*	*SD*
1.2094	3.10373	1.9524	6.02123

Means and Standard Deviations for PFAIL

SOURCE	*df*	SS	MS	F
Algorithm	3	258.8364	86.2788	2.60
Algorithm × Subject Grp	12	344.1140	28.6761	0.89
Error(Algorithm)	63	2023.3722	32.1170	

ANOVA Summary Table for PFAIL

Exhaustive		Primitive	
M	*SD*	*M*	*SD*
17.5595	22.3630	12.4999	18.1429

Local Brevity		Plan Path	
M	*SD*	*M*	*SD*
2.3810	9.0582	0	0

Means and Standard Deviations for GFAIL

SOURCE	*df*	SS	MS	F
Algorithm	3	1974.6284	658.2094	3.52
Algorithm × Subject Grp	12	4243.7423	353.6451	1.89
Error(Algorithm)	63	11765.8729	186.7598	

ANOVA Summary Table for GFAIL

Table 2: Contrast Analysis Showing Pairwise Comparison of Means

Technique	Mean	Technique	Mean	F Ratio
Exhaustive	12.880	Primitive	7.893	3.71
Exhaustive	12.880	Local Brevity	2.323	16.62*
Exhaustive	12.880	Plan Path	2.473	16.16*
Primitive	7.893	Local Brevity	2.323	4.63*
Primitive	7.893	Plan Path	2.473	4.88*
Local Brevity	2.323	Plan Path	2.473	0.00

Contrast Analysis for SFAIL

Technique	Mean	Technique	Mean	F Ratio
Exhaustive	7.849	Primitive	4.624	4.21*
Exhaustive	7.849	Local Brevity	1.209	17.85*
Exhaustive	7.849	Plan Path	1.952	14.08*
Primitive	4.624	Local Brevity	1.209	4.72*
Primitive	4.624	Plan Path	1.952	2.89
Local Brevity	1.209	Plan Path	1.952	0.22

Contrast Analysis for PFAIL

Technique	Mean	Technique	Mean	F Ratio
Exhaustive	17.559	Primitive	12.500	1.78
Exhaustive	17.559	Local Brevity	2.381	16.04*
Exhaustive	17.559	Plan Path	0.0000	21.46*
Primitive	12.500	Local Brevity	2.381	7.13*
Primitive	12.500	Plan Path	0.0000	10.88*
Local Brevity	2.381	Plan Path	0.0000	0.30

Contrast Analysis for GFAIL

* indicates significant F-value.

goals than subjects following instructions produced by the direct-translation techniques. Subjects' performance across all experimental variables shows an increase of roughly an order of magnitude, with statistically significant results obtained for both the step failure and goal failure ratios.

Since (almost all of) the results from the initial analysis are preserved when the data is adjusted to account for the length of the texts, the data suggest that the effectiveness of the cooperative techniques is not due simply to their more compact form. Not only do the cooperative approaches produce text of comparable or only slightly longer length, on average, than the Primitive algorithm (the algorithm producing the most concise texts), but the means of the Exhaustive algorithm (the algorithm producing the lengthiest texts) for all three dependent variables in the second analysis are lower than those for the Primitive algorithm. This suggests that the differences in the *content* of the descriptions produced by these techniques.

Conclusions

This paper describes the generation of textual descriptions of complex activities, specifically the generation of concise descriptions of the plans produced by computer systems. The technique, motivated by an interpretation of Grice's Maxim of Quantity, uses a generate-and-test approach to efficiently search the space of possible plan descriptions for a description that is both concise and effective.

We defined two algorithms that were used as generator functions in implementations of the CPI architecture. One, the Local Brevity algorithm, selects candidate descriptions based on the importance that the elements of a description hold for the hearer's comprehension of the description as indicated by psychological studies. The other, called the Plan Path algorithm, selects candidates based on the processing that was used to produce the source plan. For both implementations, a common evaluator function was employed that used a domain-independent planning algorithm as the hearer model's planning system and a domain-independent approach to applying the model to determine the acceptibility of candidate plan descriptions.

To characterize the efficacy of the two generator functions relative to one another and relative to two alternative direct-translation algorithms, we performed a task-efficacy evaluation. In this experiment, subjects that followed instructions produced by the CPI algorithms committed fewer execution errors and achieved more of their tasks' top-level goals than subjects following instructions produced by other techniques. The experimental results provide clear support for the greater efficacy of the cooperative techniques.

Acknowledgements

Support for this work was provided by the Office of Naval Research, Cognitive and Neural Sciences Division (Grant Number N00 014-91-J-1694) and from the DoD FY92 Augmentation of Awards for Science and Engineering Research (ASSERT). The author thanks Johanna Moore and Martha Pollack for many helpful discussions and David Allbritton for his advise on experimental design.

References

Dale, R., and Reiter, E. 1995. Computational interpretations of the Gricean Maxims in the generation of referring expressions. *Cog. Science* 19(2):233–263.

Golding, J.; Graesser, A.; and Millis, K. 1990. What makes a good answer to a question? testing a psychological model of question answering in the context of narrative text. *Discourse Processes* 13:305–326.

Graesser, A.; Roberston, S.; Lovelace, E.; and Swineheart, D. 1980. Answers to why questions expose the organization of story plot and predict recall of actions. *J. of Verb. Learning and Verb. Behavior* 19:110–119.

Grice, H. P. 1975. Logic and conversation. In Cole, P., and Morgan, J. L., eds., *Syntax and Semantics III: Speech Acts*. New York, NY: Academic Press. 41–58.

Hartley, A., and Paris, C. 1997. Multilingual document production: from support for translating to support for authoring. *Machine Translation, Special Issue on New Tools for Human Traslators* 12(1-2):109–129.

Kambhampati, S.; Knoblock, C.; and Qiang, Y. 1995. Planning as refinement search: a unified framework for evaluating design tradeoffs in partial-order planning. *Artificial Intelligence* 76:167–238.

Kintsch, W., and Van Dijk, T. A. 1978. Propositional and situational representations of text. *Psychological Review* 85:363–394.

Mellish, C., and Evans, R. 1989. Natural language generation from plans. *Computational Linguistics* 15(4):233 – 249.

Penberthy, J. S., and Weld, D. 1991. UCPOP: A sound, complete partial order planner for ADL. In *Proceedings of the Third International Conference on Knowledge Representation and Reasoning.*

Rumelhart, D. E. 1977. Understanding and summarizing brief stories. In LaBerge, D., and Samuels, S. J., eds., *Basic processes in reading: perception and comprehension*. Erlbaum.

Tate, A. 1977. Generating project networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 888 – 893.

Trabasso, T., and Sperry, L. 1985. Causal relatedness and importance of story events. *J. of Memory and Language* 24:595–611.

van den Broeck, P. 1988. The effects of causal relations and hierarchical position on the importance of story statements. *J. of Memory and Language* 27:1–22.

Vander Linden, K., and Martin, J. H. 1995. Expressing rhetorical relations in instructional text: A case study of the purpose relation. *Computational Linguistics* 21:29–57.

Walker, M., and Moore, J. 1997. Empirical studies in discourse. *Computational Linguistics* 23(1):1–12.

Wright, D., and Hull, P. 1990. How people give verbal instructions. *J. of Appl. Cog. Psych.* 4:153–174.

Young, R. M.; Pollack, M. E.; and Moore, J. D. 1994. Decomposition and causality in partial order planning. In *Proceedings of the Second International Conference on AI and Planning Systems*, 188–193.

Young, R. M. 1996. Using plan reasoning in the generation of plan descriptions. In *Proc. of the National Conference on Artificial Intelligence*, 1075–1080.

Young, R. M. 1997. *Generating Descriptions of Complex Activities*. Ph.D. Dissertation, Intelligent Systems Program, University of Pittsburgh.