The Less Obvious Side of NONLIN

Austin Tate 22-Aug-83

This working paper has been prompted by recent work on hierarchic non-linear planners (NONLIN is such a system) and on some work in temporal logics to support planning. The only easily available reference for NONLIN was an IJCAI-77 (Tate, 1977) paper which concentrated on one particular feature. The complete description is available in an Edinburgh Department of Artificial Intelligence Research Report (Tate, 1975). The NONLIN features to be highlighted below will be related to the different terminology used by other workers and I will show where their systems have gone beyond the applicability of NONLIN.

All of the work on hierarchic non-linear planners has a root in Sacerdoti's landmark work on the NOAH planner at SRI (Sacerdoti, 1975). Work in Edinburgh on an U.K. Science and Engineering Research Council grant "Planning: a joint AI/OR approach" used NOAH and work on using "Goal Structure" to direct the search of planners as the basis for the NONLIN planner used on the project. The application domain was the production of project networks for civil engineering construction and maintenance tasks (e.g., house building and electricity turbine overhaul).

This paper will start by relating the feature of NONLIN that will be familiar to most workers in the planning field (through the IJCAI-77 paper) to the underlying principles of NONLIN. The paper will then give an introduction to some of the other features of NONLIN which may be of interest for others working in the field.

Correcting for Interactions between Parallel Branches of the Plan

This was the theme of the IJCAI-77 paper (Tate, 1977) and the most often referenced feature of NONLIN by other authors. The paper described the method by which NONLIN extended the search space of Sacerdoti's NOAH to ensure that detected interactions were corrected for (NOAH only tried one of two legal plan re-orderings).

The analysis of correcting for the interactions was based on earlier work on a linear planner INTERPLAN (Tate, 1974). This had its own roots in the HACKER (Sussman, 1973) mechanisms for debugging PROTECTION-VIOLATION faults during program synthesis. It proved the case that the same analysis of the solution space applied to non-linear planning systems also. In fact, the interaction detection and correction features built a table of the underlying "goal structure" of the emerging parallel plan in NONLIN (see the section on Goal Structure below) and can thus discriminate actual logical interactions from unimportant side effects. This was the subject of my thesis work (Tate, 1975) as it related to linear planners.

Though I did not realise it at the time, the STRIPS MACROPS operator tables format for sub-plan re-use (Files, Hart and Nilsson, 1972) hold most of the required information, but to my belief they were not used in

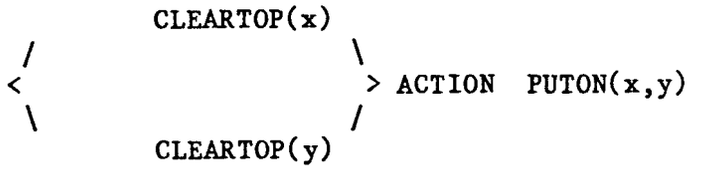
this way. Some recent works in A.I. planning are converging on a similar ability to discriminate between genuine and unimportant interactions but in their comparisons with earlier work usually only relate their systems to NOAH which was very weak in this respect.

Goal Structure

This was considered by us to be one of the most important contributions of NONLIN (and earlier work on INTERPLAN). A Goal Structure table was kept during planning to record what facts had to be true at any point in the network and the possible "contributors" that could make them true. The system was able to plan without choosing one of the (possibly multiple) contributors until this was forced by interaction detection. This table was used to detect important interactions (conveniently ignoring unimportant side effects) and could be used to directly suggest the 2 minimal re-linking orders which could overcome each interaction (the subject of the IJCAI-77 paper).

However, there had to be a way of being more precise about the actual facts needed at any point in the plan. For normal "GOAL" nodes which occur in non-linear planners operators (I now prefer to think of these as "achieve" preconditions), these were precisely specified by the planner, as only the relevant effect of an operator which achieved the goal was recorded as necessary to maintain the Goal Structure. Other effects (if not used elsewhere) could be contradicted without the planner worrying.

To properly maintain the intended Goal Structure of a plan, it is important to realise that there is also internal Goal Structure in the operators specified by a user to describe the domain. This must be properly reflected in the plan's Goal Structure whenever operators are used. When a user gives a blocks world operator expansion such as:



there is an implicit assumption that once CLEARTOP(x) is achieved for the relevant instantiation of the variable x, that this is needed from that point at least until the PUTON ACTION that follows it. The same goes for CLEARTOP(y). If PUTON(x,y) is itself expanded to lower levels of detail, the CLEARTOP(x) and CLEARTOP(y) conditions may be needed up to particular points within such an expansion.

This was dealt with by providing a Task Formalism (operator specification) language for domains which allowed a precise encoding of the Goal Strucure ranges within any exapnsion (subplan) for the operator. Sensible default handling of such condition ranges meant that the normally implied structures with GOAL and subsequent ACTION nodes in an expansion could be automatically inserted by the Task Formalism language interpreter. However, the Task Formalism allowed a greater degree of expression of domain specific information where this was known. See the section on Task Formalism below for more details.

Wilkins' ability to specify a "Plan Rationale" in SIPE operators (Wilkins, 1983) and use this in interaction detection can be seen as a form of the more general Goal Structure mechanisms used in NONLIN. The SIPE ability to specify Plan Rationale to give the important effect of a subplan (in terms of the major goal to be achieved) was to overcome earlier problems with the NOAH scheme of assuming that all the effects specified in an add list of an operator are achieved at the end of the last node of the subplan specified and that all the effects are important preconditions to allow the following actions to be performed. However, the SIPE scheme of only searching for interactions on the Plan Rationale pattern is probably a move too far in the other direction.

Other authors have referred to underlying structures for their planners that seek to achieve the same benefits as the Goal Structure of NONLIN. I believe that Goal Structure will be important for work on plan execution monitoring and constructive re-planning schemes. Such work is being persued in Edinburgh at present.

Table of Multiple Effects (TOME) in NOAH

NOAH used a TOME to record all the world model changes at any node. The NOAH TOME was modelled on earlier work in the INTERPLAN linear planner on classifiers (called "ticklists") for categories of interaction problems (Tate, 1974). Parallel nodes with contradictory effect values were sought in the TOME by NOAH plan critics to detect interactions. Nodes parallel to a GOAL which achieved it were also sought as candidates for linking to use these, so called, beneficial side-effects. However, the NOAH scheme detected and attempted to correct for many interactions which were irrelevant to successful plan execution due to the assumption that all effects of an operator were important. It thus excluded a large class of solvable problems, especially where the domain was represented realistically (with large numbers of effects).

NONLIN did include a data structure called the TOME, but it should more properly have been called a TOE (Table of Effects). This was not used for interaction detection and correction (the Goal Structure was used for this), but for question answering in the partially ordered network of nodes in the plan.

Question Answering in a Partially Ordered Network of Actions

A.I. planning languages such as CONNIVER (McDermott and Sussman, 1972), and data base systems to support planners such as HBASE (Barrow, 1975) have provided efficient mechanisms for storing facts with respect to a tree of "contexts". Each "context" is made up of a series of differential changes (called "layers") to some initial state. Branching gives the effect of different final states in which questions can be asked. In a non-linear planner, questions must be asked as to what is true at any point in a network of nodes. There is an analogy here to systems which provide property inheritance with multiple ancestors.

NONLIN introduced an algorithm for asking questions in such a network which could be based upon the storage of facts in a tree of contexts (to

make use of the existing mechanisms avaliable) supplemented by the Table of Effects and a graph search algorithm on the network plan structure itself. This is fully documented in Tate (1976) and a more efficient version, which was not implemented, is descibed in Daniels and Tate (1982). NONLIN used the HBASE semantic network data base package augmented with context facilities as described above; if-added, if-removed and if-needed theorems as used in CONNIVER to add some measure of deductive capabilities to the world modelling; and a new variable type (labelled $*) for the HBASE pattern matcher which behaved similar to the PROLOG logical variable (it could take on simple restrictions before receiving a definitive value, and on failure could be matched to alternative legal values).

The representation of the facts true at points in a plan recognised that invariant facts (true in any state) could be handled more efficiently than normal changing information. Hence, the facts always true were kept at the root of the tree of contexts (in a special context called "ALWAYS-CONTEXT"). This was consulted before invoking the normal question answering system to improve prerformance. A similar scheme which also adds a type/property inheritance hierarchy is included in Wilkin's SIPE planner.

Search Control Strategy - Heuristic and Dependency-directed

Most authors have assumed that a standard depth-first backtracking algorithm was employed in NONLIN to achieve the proper handling of interactions in a NOAH-like planner. The IJCAI-77 paper (Tate, 1977) which described the technique did not assume any particular control regime as it was intended as a simple-to-describe "fix" that anyone could add to NOAH-like planners of many types then being worked on at various laboratories.

The actual scheme in the original NONLIN used a heuristic search tree of alternative solution spaces (representing different ways to tackle the problem and the level of detail of the network up to various points). It had a simple heuristic evaluator which preferred major choice points such as alternative linearisations to correct for interactions, choices of different operators, etc., to minor choice points such as choosing alternative instantiations for a variable (e.g., choosing alternative instances of a block to perform some action after the first chosen block proves unsuitable). The basic heuristic search regime would of course support heuristic evaluators of greater complexity but this was not a particular concern of the research. However, see the section below on planning with time and cost information for one experiment that was carried out.

Daniels (1977) added a dependency-directed control structure to NONLIN in which a separate "decision graph" was constructed to record the interrelationships between choices and their effects on the emerging plan. On failure, selective undoing of plan parts and their replacement with appropriate alternatives was possible. The decision graph was based on earlier Edinburgh work on a travel planning and re-planning system (Hayes, 1975).

Task Formalism

1. Declarative The problem domain is described to the NONLIN planner through a completely declaarative language called the "Task Formalism". This contrasted with the procedurally specified (SOUP code) operator descriptions in NOAH. NONLIN returned to STRIPS-like descriptions. The TF was parsed by a simple front end and used macro features to allow for semantically suggestive synonyms for some of the keywords. The interface to the planner was an internal format which more closely reflected the structures used to represent plan networks and used to index on selecting appropriate information as necessary.

2. User Interfaces and Applicability The TF is fully described in Tate (1976). It went beyond the immediate needs of the research project for which NONLIN was used ("Planning: a joint A.I./O.R. approach") and was intended to form a reasonably stable base for that project and for later application and research work. Hence the constructs were intended to allow for the embedding of significantly more heuristic information about the domain than had been the case with earlier planners. The TF also reflected the underlying organising principle of Goal Structure to describe the "intent" of the sub-plan specified for any operator.

   Although the TF parser actually used with NONLIN was very simple, the TF language was designed as an intermediate formalism to which a user front-end with a graphical interface and knowledge based assistance could be added. The design of such an interface is now underway at Edinburgh.

3. Typed Conditions for Planning The TF operator descriptions allowed the specification of "types" for the preconditions. Earlier planning language work had identified the problem with the single precondition type which could either be already true or could induce subgoaling to be made true. The POPLER (Davies, 1973) system introduced 2 precondition types into the PLANNER-like languages - one to simply look up if something was true (perhaps instantiating variables), and the other which was allowed to recursively make the condition true if it did not already hold. This overcame important inefficiencies in problem solving with the PLANNER language (Hewitt, 1972).

   TF extended this notion and mated it with a "process" oriented view of operator descriptions. A TF operator description specifies a method by which some higher level action can be performed (or higher level goal achieved). The operator introduces lower level actions under the control of the operator "manager" (these are his own resources in some sense). He says that something is to be done in order to achieve part of his job. In TF these are specified as SUPERVISED conditions. The "manager" also relies on other agents to perform tasks that are their own responsibilities, but affect the ability of this manager to do his job. These are given as UNSUPERVISED conditions. There are other conditions which the "manager" may wish to impose on the applicability of particular solutions (i.e., don't bother to try this method for house building

if the building is over 5 stories tall). These were termed HOLDS and USEWHEN conditions in versions of NONLIN, but I now believe some other type name such as ONLYIF may reflect the semantics better. Later releases of Vere's DEVISER planner added an *ALREADY condition which performs the same function.

NONLIN in fact had 2 other condition types which really relate to other planning abilities and are not strictly part of the process specification aim of the TF. These were QUERY for optional inclusion of sub-plan parts, and COMPUTE to communicate with externally defined computations and data bases. The intensive literals and VALUE.OF mecahism in Vere's DEVISER planner perform a similar function to the TF COMPUTE "condition".

The TF parser had a facility for automatically filling in SUPERVISED conditions between a GOAL node and the following ACTION nodes. This ability allowed it to take operator descriptions in the form accepted by most A.I. planners and still produce the necessary TF condition structures without the user having to be concerned with this. However, this could be overridden on any individual operator for the inclusion of more precise domain specific knowledge.

The TF condition types provided important search information for the planner, facilitated the Goal Structure based interaction detection and correction facilities and were used to reduce the alternatives which needed to be considered when using the dependency-directed backtracking system added to NONLIN (Daniels, 1977). In retrospect, it seems that the issue of typed preconditions and their applicability to the various stages of knowledge explication, planning and search space control should receive more attention. The points covered briefly above will be the subject of a more extensive paper later.

4 Variable Restrictions The operator descriptions in TF could specify that variable should be "restricted" to only match certain values. The restriction was specified as a general pattern matcher pattern interpreted by the underlying HBASE data base package. It could express logical connectives, invoke any general matching function, etc. The notion of constraining variables in operators during planning has been greatly extended in Vere's DEVISER and Wilkin's SIPE. These systems allow constraints to be accumulated rather than forcing a choice between alternatives which satisfy the constraint early in the planning process (as NOAH and NONLIN would). This helps keep the planner search space from growing too large.

5. Deductive World Modelling If-added, if-removed and if-needed theorems (as in CONNIVER) were added to the underlying HBASE data base system used in NONLIN. This and the ability to specify a "deductive" operator (no actions, just goals or conditions and effects) allow deductive world models to be specified.

6. Main Effects A simple facility to provide a mechanism for seeing the effects of choosing any of a set of similar operators applicable to the expansion of any pattern is provided. This allowed the common

effects of such operators to be separately specified. The planner could detect conflicts and benefits of the set of operators as a whole before being committed to choosing any particular one. The operator descriptions themselves then gave the effects unique to their individual methods of performing the task. This has advantages from the point of view of reducing the search space and increasing the understandability of a large operator set in terms of sets of related methods. This concept has some similarity to the "teams" of operators used in other systems to restrict the range of operators to be tried in a given situation.

Expansions and "compiled" TF Forms

The TF parser translated the declarative representation of a sub-plan (or "expansion") for an operator into an internal form that was very closely related to the internally stored form of an actual plan network being constructed. The internal links in the sub-plan were based on pairs of sub-plan node numbers with a zero base. Any actual plan was associated with a variable, MAXNODES, that represented the number of nodes in the plan at each stage. When an expansion was chosen, it could be spliced into the plan by adding most of the sub-plan node list onto the end of the full list of nodes in the plan and adding MAXNODES to the numbers associated with the sub-plan nodes. A simple "relocation" process on internal relationships such as the ordering links, condition attachment points, etc could then fix up the sub-plan details to their proper relationship to the overall plan network. The last node of an expansion actually replaced the higher level node being expanded and inherited its node number. This meant that many internal relationships would already be correctly attached. This is fully described in Tate (1976).

Planning with Time or Cost Information

The Task Formalism allowed a "duration" to be associated with each action in the operator hierarchy. An option available when NONLIN was entered brought into play a system that computed Earliest Start Time (EST) and Latest Start Time (LST) information for each node in the network at any stage of planning. The system also computed the total plan duration (finding the critical path nodes as it did so - actions which if delayed would cause the entire plan to take longer to execute).

The NONLIN planner had incremental algorithms to propogate the EST and LST values through the network as expansions were made and links introduced. There appears to have been little work in Operational Research on such "incremental" algorithms for maintaining such critical path analysis data (in O.R. the plan network is often specified in advance of using analysis algorithms).

Daniels (1977) described a version of NONLIN in which choices of alternative operators and alternative orders of linking to correct for interactions were dependent on a plan cost measure which was the sum of the costs of the nodes in the plan added to some factor of the overall plan duration. She termed this the plan "efficiency". This was used to make more sensible heuristic choices of alternative search branches. The process is also documented in Daniels and Tate (1982).

Vere's DEVISER planner (Vere, 1981) has taken the ability of hierarchic non-linear planners to cope with time information much further. His planner adds the capability to actually specify an EST and LST for any node in the plan (rather than these simply being assumed to be dictated by the ordering of the nodes as in NONLIN). Vere's incremental algorithms for propogating EST and LST information through the plan network can account for these "time window" specifications and can signal when they are violated. Vere also allows externally caused scheduled events to be taken into account during planning and allows for delayed effect events (caused by earlier actions) to be handled (e.g., a spacecraft is steady some time after firing a thruster).

Interactive Features

NONLIN was able to operate in "stand-alone" mode where the Task Formalism was completed in a file in advance and given to the TF parser. The problem was specified to NONLIN using the same TF parser and the system would work on the problem through to completion. This was the mechanism most often used with the system. However, if NONLIN tried to expand some pattern for which there were no operators, the system would advise the user and make the TF parser available for on-line entry of further TF forms. After a problem had been solved in such an incremental specification mode, the external form of the TF given during the session could be output to a file.

NONLIN was also used on a large problem of electricity turbine maintenance overhaul in which the complete solution to the problems set was unrealistic for the NONLIN planner as implemented. Hence, a QUERY condition type was added by means of which the user could interactively decide whether some sub-action of an expansion was to be included or not.

The decisions at choice points in the NONLIN system were reported to the user and were localised in the NONLIN code in order that a dialogue could take place on which choices to prefer. However, such a dialogue system was not implemented. It was believed to be particularily important for giving the planner assistance with choosing instances for variables where these were not obviously constrained.

Wilkin's SIPE planner has taken a more interactive approach to cooperative man/machine plan construction and allows the user to intervene at many points in the planning process.

Application Areas

The NONLIN system was used on various block stacking tasks intended to test the features of the planner on problems known to exhibit difficulties for planning systems. House building tasks in a simple formalism that usually led to 25-50 node networks being generated was used for much of the early work. A larger problem domain was coded in TF for electricity turbine overhauls. This had 3 levels in the hierarchy, with 70, 300 and 750 nodes at the various levels. It was used to construct plans with around 300-400 nodes. However, this domain

should be considered "simple" in terms of the interaction problems and number of choices presented (hence it required little or no branching in the search space).

References

Barrow, H.G. (1975) "HBASE: a fast clean efficient data base system" D.A.I. POP-2 library documentation.

Daniels, L. (1977) "Planning: Modifying non-linear plans" D.A.I. working paper no.24.

Daniels, L. and Tate, A. (1982) "A retrospective on the 'Planning: a joint AI/OR approach' project" D.A.I. working paper no.125.

Davies, D.J.M. (1973) "POPLER 1.5 Reference Manual" D.A.I. Theoretical Pshychology Unit Report no.1.

Fikes, R.E., Hart, P.E. and Nilsson, N.J. (1972) "Some new directions in robot problem solving" in "Machine Intelligence 7", Meltzer, B. and Michie, D., eds., Edinburgh University Press.

Hayes, P.J. (1975) "A representation for robot plans" Advance papers of IJCAI-75, Tbilisi, USSR.

Hewitt, C. (1972) "Description and Theoretical Analysis (using schemata) of PLANNER" M.I.T. A.I. Lab. Memo no.MAC-TR-256.

McDermott, D.V. and Sussman, G.J. (1972) "The CONNIVER Reference Manual" M.I.I. A.I. Lab. Memo no.259.

Sacerdoti, E.D. (1975) "The non-linear nature of plans" Advance papers of IJCAI-75, Tbilisi, USSR.

Sussman, G.A. (1973) "A computational model of skill acquisition" M.I.T. A.I. Lab. Memo no.AI-TR-297.

Tate, A. (1974) "INTERPLAN: a plan generation system which can deal with interactions between goals" Machine Intelligence Research Unit Report no.MIP-R-109.

Tate, A. (1975) "Using Goal Structure to direct search in a problem solver" Ph.D. Thesis, Edinburgh University.

Tate, A. (1976) "Project planning using a hierarchic non-linear planner" D.A.I. Research Report no.25.

Tate, A. (1977) "Generating Project Networks" Proceedings of IJCAI-77, Boston, USA.

Vere, S. (1981) "Planning in time: windows and durations for activities and goals" NASA Jet Propulsion Lab. Technical Report.