N88-16416

Planning and Scheduling for Robotic Assembly†

Barry R. Fox
Artificial Intelligence Group
McDonnell Douglas Research Laboratories
PO Box 516, St. Louis, MO 63166

Abstract. This paper describes a system for reasoning about robotic assembly tasks. The first element of this system is a facility for itemizing the constraints which determine the admissible orderings over the activities to be sequenced. The second element is a facility which partitions the activities into independent subtasks and produces a set of admissible strategies for each. Finally the system has facilities for constructing an admissible sequence of activities which is consistent with the given constraints. This can be done off-line, in advance of task execution, or it can be done incrementally, at execution time, according to conditions in the execution environment. The language of temporal constraints and the methods of inference presented in related papers are summarized. It is shown how functional and spatial relationships between components impose temporal constraints on the order of assembly and how temporal constraints then imply admissible strategies and feasible sequences.

Introduction

Construction of the Space Station requires an unprecedented degree of advanced planning and scheduling. The sequence of construction activities must be consistent with the functional and spatial relationships between the components of the space station, it must be consistent with feasible methods for delivering and fabricating the structure, and it must be consistent with high-level schedules and timetables. It must be possible to produce schedules which satisfy these constraints and it must be possible to quickly produce revised schedules in the event of problems and delays.

In this context the problem is not simply to produce a single schedule of construction activities, but rather to carry the production plan through multiple levels of refinement. In its most general form the plan may be simply a collection of the constraints which must be satisfied. After some refinement the plan may be partitioned into independent subtasks and the plans for those subtasks may be factored into strategies. Ultimately, all of the construction activities will be mapped onto intervals of a timeline. When problems or delays occur it should be possible to revise the schedule according to the constraints imposed by a higher level in the scheduling hierarchy. This view of planning and scheduling is supported by a system for reasoning about robotic assembly tasks described below.

† Research supported in part by the McDonnell Douglas Independent Research and Development Program.

The Representation of Robotic Assembly Tasks

Robotic assembly is fundamentally a serial process. Although the steps of an assembly can be ordered in a variety of ways, a robot typically performs an assembly one step at a time. This has two important implications. First, the analysis of assembly tasks is simplified. Problems of synchronization and concurrency are found only in the interaction between processes not within the assembly task itself. Second, the execution of assembly tasks is simplified. The flexibility in the ordering of the assembly steps can be exploited to adapt the sequence of operations to execution time conditions [Fox and Kempf, 1986a]. However, the ability to reason about the execution of an assembly task depends intimately upon the form and content of some abstract representation of that task. Proposed representations include state transition networks [Whitney], and-or graphs [deMello and Sanderson], precedence diagrams [Prenting and Battaglin], Petri nets [Drummond],[Malcolm and Fothergill], and temporal constraints [Fox and Kempf, 1986b]. These and other representations are discussed in detail in another paper [Fox, 1987a]. The remainder of this paper will focus on the derivation and analysis of temporal constraints.

The Derivation of Temporal Constraints

Allen defines 13 relationships that can exist between two intervals of time [Allen]. However, the serial execution of an assembly task T reduces the relationship between any two steps to exactly two possibilities: $\forall x, y \in T, (x < y) \vee (y < x) \wedge \neg((x < y) \wedge (y < x))$ (serial axiom) where the expression $(x < y)$ denotes the constraint that step x must strictly precede step y. In addition, the transitive nature of time determines the effect of combinations of relationships: $\forall x, y, z \in T, (x < y) \wedge (y < z) \implies (x < z)$ (transitive axiom).

The constraints which govern the execution of assembly task can be stated as a conjunction of the relationships which exist between steps of the task. In some cases the order of two steps is unconstrained and it is unnecessary to explicitly state their relationship. It is simply assumed that their order of execution is governed by the serial axiom. In other cases exactly one of the two possible orderings is required and it is sufficient to state that relationship as a primitive constraint of the form $(x < y)$. In more complicated cases the necessary ordering over two steps will be conditional upon the ordering of other steps. In such situations it will be necessary to state the relationships among these steps using combinations of primitive constraint expressions and the logical operators $\wedge$ (and), $\vee$ (or), $\neg$ (not), and $\implies$ (implies).

The underlying physical relationships which determine the necessary temporal constraints are varied. While not exhaustive, a number of cases will be enumerated in order to suggest the range of possibilities.

Some of the relationships between the objects to be manipulated determine enabling conditions for steps of the assembly task. For example, the installation of a major component of an assembly then enables the installation of those parts that are attached to or supported by that part. In other situations, the necessary support for a part may be a collection of other parts. In more complex cases there may exist multiple distinct strategies or methods for completing an assembly and a given step may be enabled by different conditions under each strategy.

In a similar fashion, some of the relationships between the objects to be manipulated determine disabling conditions for steps of the assembly task. In some cases the installation of a part may prevent the subsequent installation of other parts. In other cases, the installation of a part may be prevented by the installation of any one of several other parts. Occasionally the destination for a part can be approached from several different directions and the installation of that part will be disabled only after every one of several other parts has been installed.

It is not unusual for some steps of the assembly to be temporarily disabled. For example, the installation of a special jig may be necessary for some stage of an assembly but it may interfere with some otherwise admissible operations. Those operations must be performed either before the installation of the jig or after its removal but they are impossible while it is in place. In other cases the assembly may go through fragile or unstable states when forceful or agitating operations should be avoided. Although they may be otherwise enabled, the rough operations should either precede or follow all of the delicate operations.

The specification of an assembly task begins with an analysis of the objects to be manipulated and the operations to be performed. That analysis will reveal the significant physical relationships which enable and disable the various operations. The result of that analysis will be a logical formula composed of primitive ordering constraints of the form $(x < y)$ and the logical operators $\wedge$ (and), $\vee$ (or), $\neg$ (not), and $\Longrightarrow$ (implies). Example assembly tasks are presented elsewhere [Fox 1987a],[Fox and Kempf, 1986b].

While this analysis may be a human activity, the translation of physical relationships into temporal constraints can be automated to a certain degree. Common physical relationships can be catalogued along with methods for translating them into temporal constraints. The catalogue implemented by the author includes the generic relationships *enabled-by*, *disabled-by*, and *prohibited-by* as well as more specific relationships such as *supported-by*, and *attached-to*. These are translated into temporal constraints by simple macro substitution.

The specification process is not without difficulty. It is possible that a significant relationship among the parts or operations to be performed will be encountered which has not been previously catalogued. In that case the analyst must directly produce a formula which defines the admissible ordering of the operations involved. There is also the possibility that the analyst will overlook some significant constraint or erroneously include some unnecessary constraint. Some of these difficulties can be avoided by deriving temporal constraints from an analysis of certain significant states of the given task [Fox, 1987b].

The Derivation of Subtasks, Strategies, and Sequences

Given a task composed of a set of activities and a temporal constraint expression which governs their execution, it is quite simple to partition the activities into independent subtasks. First, extract all of the primitive constraints that occur in the given constraint expression (regardless of the logical structure of the formula). Then, for each primitive constraint $(x < y)$, construct the constraint $(x \bowtie y)$ which denotes the fact that x *is related to* y. The relation $\bowtie$ is reflexive, symmetric, and transitive, thereby satisfying the requirements of an equivalence relation. Next, form the closure of the equivalence relation defined by $\bowtie$. Each of the resulting equivalence classes contains only activities that are related by some combination of ordering constraints. Moreover, there are no ordering constraints which cross the boundaries of the equivalence classes. Hence, the execution of the given task can be treated as the interleaved execution of multiple independent subtasks as defined by the equivalence classes.

Given a task composed of a set of activities and a temporal constraint expression which governs their execution, it is a non-trivial task to construct a feasible execution sequence. It is simple to demonstrate that the problem of determining the satisfiability of an arbitrary temporal constraint expression is NP-Complete and the process of determining an admissible first step is NP-Hard. However, some temporal constraint expressions are easy to analyze. Any task which can be represented by a conjunction of primitive constraints (i.e., a strict partial order) can be analyzed and sequenced by simple polynomial time algorithms. Although not every task can be reduced to such a simple representation, every task can be represented as the union of a set of conjunctive plans (i.e., a disjunction of conjunctions of primitive constraints) which together cover every admissible sequence of operations. Most algorithms which can be applied to conjunctive constraint expressions can be adapted to the more general disjunctive normal form.

Given an arbitrary temporal constraint expression, an equivalent disjunctive normal form constraint expression can be produced by purely algebraic means. Since this is an NP-Complete language, there are two hidden perils associated with this process. If simple, direct methods are used to create the disjunctive normal form, then the size of the resulting expression can be prohibitively large; if more sophisticated methods are used to produce the smallest possible disjunctive normal form, then the time required may be prohibitive. For many problems, careful use of heuristic methods reported previously [Fox 1987a] yield a reasonably small disjunctive normal-form constraint expression in a reasonable amount of time.

The disjunctive normal form of a given temporal constraint expression has several important interpretations. Like the original constraint expression, it still represents the task to be accomplished; it still represents the constraints over the the individual operations to be performed; and it still implicitly represents the set of admissible sequences for performing those operations. The added interpretation that can be applied to the disjunctive normal form is that each conjunctive clause can be interpreted as a *strategy* for performing the given task. This can be particularly useful to engineers responsible for designing and managing the execution of complex tasks. They can first focus on the constraints over the constituent operations. Then they can verify and refine the specification of the task by analyzing the admissible strategies which are produced by this normalization. Methods for analyzing the resulting strategies are presented elsewhere [Fox, 1987a].

The normalization and analysis described above serves only one purpose: the production of a set of constraints which circumscribe the admissible sequences over a set of operations and the transformation of those constraints into a form suitable for processes of sequencing. The sequence of operations can be determined off-line, prior to execution time, according to expected run-time conditions; or on-line, at execution time, according to actual run-time conditions.

A feasible sequence of operations under a single strategy can be derived from the process of pebbling the nodes of a directed acyclic graph derived from that strategy. Each step of the task is translated into a node of the graph and each primitive ordering constraint ($x < y$) is translated into a directed arc from node x to node y. Pebbling is governed by a single rule: a node can be pebbled only if all of its predecessor nodes have been previously pebbled. An initial node, with no predecessors, can be pebbled at any time. By analogy, the set of steps that can be executed next in some state of the task correspond exactly to the set of nodes that can be pebbled next in the analagous state of the graph. The process is complete when all the nodes are covered. The set of all possible pebbling sequences is identical to the set of all possible execution sequences for that strategy. Simple methods have been developed which extend the pebbling analogy to the execution of a task composed of multiple strategies [Fox, 1987a].

Conclusion

A method for representing robotic assembly tasks based upon temporal constraints has been presented. Methods for deriving those constraints from enabling and disabling physical relationships have been summarized along with methods for factoring a task into independent subtasks, strategies, and feasible sequences of operations.

References

[1] Allen, J., Maintaining knowledge about temporal intervals, *Communications of the ACM* 26, pp. 832-843, 1983.

[2] deMello, L.S.H. and Sanderson, A., And/Or graph representation of assembly plans, in *Proceedings Fifth National Conference on Artificial Intelligence*, Philadelphia, Penn., pp. 1113-1119, 1986.

[3] Drummond, M., Plan Nets: a formal representation of action and belief for automatic planning systems, Ph.D. dissertation, Department of Artificial Intelligence, University of Edinburgh, 1986.

[4] Fox, B.R., A representation for finite serial processes, Ph.D. Dissertation, Department of Computer Science, University of Missouri, Rolla, MO, 1987.

[5] Fox, B.R., The derivation of temporal constraints, submitted to 1988 IEEE International Conference on Robotics and Automation, Philadelphia, PA, April 1988.

[6] Fox, B.R. and Kempf, K.G., Opportunistic scheduling for robotic assembly, in *Industrial Engineering, Selected Readings* E.L. Fisher and O.Z Maimon (eds), Institute of Industrial Engineers, Atlanta, GA, 1986.

[7] Fox, B.R. and Kempf, K.G., A representation for opportunistic scheduling, in *Third International Symposium on Robotics Research*, O. Faugeras and G. Giralt (eds), MIT Press, Cambridge, MA, 1986.

[8] Malcolm, C. and Fothergill, P., Some architectural implications of the use of sensors, in *Intelligent Autonomous Systems*, L.O. Hertzberger and F.C.A. Groen (eds), North-Holland, Amsterdam, 1987.

[9] Prenting, T.O. and Battaglin, R.M., The precedence diagram: a tool for analysis in assembly line balancing, *Journal of Industrial Engineering*, vol 15, #4, pp. 208-211, July-August 1964.

[10] Whitney, D.E., State space models of remote manipulation tasks, *IEEE Transactions on Automatic Control*, vol AC-14, #6, Dec. 1969.