R1: A Rule-Based Configurer of Computer Systems*

John McDermott

Department of Computer Science, Carnegie-Mellon University, Schenley Park, Pittsburgh, PA 15213, U.S.A.

Recommended by Edward A. Feigenbaum

ABSTRACT

R1 is a program that configures VAX-11/780 computer systems. Given a customer's order, it determines what, if any, modifications have to be made to the order for reasons of system functionality and produces a number of diagrams showing how the various components on the order are to be associated. The program is currently being used on a regular basis by Digital Equipment Corporation's manufacturing organization. R1 is implemented as a production system. It uses Match as its principal problem solving method; it has sufficient knowledge of the configuration domain and of the peculiarities of the various configuration constraints that at each step in the configuration process, it simply recognizes what to do. Consequently, little search is required in order for it to configure a computer system.

1. Introduction

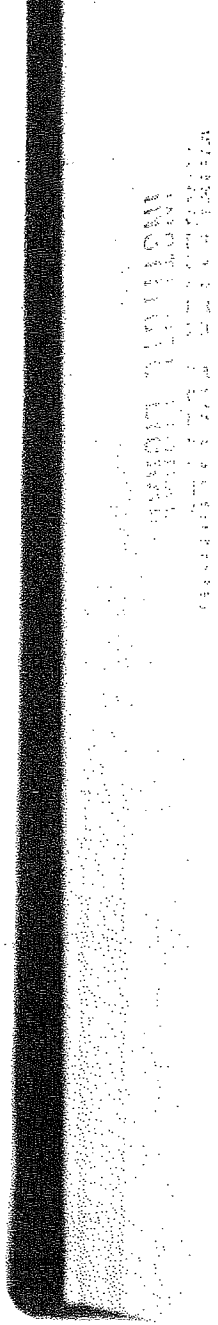
R1[1] is a rule-based system that has much in common with other domain-specific systems that have been developed over the past several years [1, 13]. It differs from these systems primarily in its use of Match rather than Generate-and-test as its central problem solving method; rather than exploring several hypotheses until an acceptable one is found, it exploits its knowledge of its task domain to generate a single acceptable solution. R1's particular area of expertise is the

*The development of R1 was supported by Digital Equipment Corporation. The research that led to the development of OPS4, the language in which R1 is written, was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, and monitored by the Air Force Avionics Laboratory under Contract F33615-78-C-1151. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of Digital Equipment Corporation, the Defense Advanced Research Projects Agency, or the U.S. Government. VAX, PDP-11, UNIBUS, and MASSBUS are tradmarks of Digital Equipment Corporation.

[1] Four years ago I couldn't even say "knowledge engineer", now I . . .

Artificial Intelligence **19** (1982) 39–88

0004-3702/82/0000-0000/$02.75



- The customer's order must be determined to be complete; if it is not, what components are missing must be added to the order.
- The spatial relationships among all of the components (including those that added) must be determined.

The criterion of success for whether a configuration is complete does not re in any simple test, but involves instead particular knowledge about all individual components and their relationships. The criterion of succes spatial arrangement is more compact (reflecting the uniform characte geometric structure), but it too involves particular knowledge on a compor by component basis. Thus, task accomplishment is defined by a large se constraints embodying a large amount of knowledge.

R1's approach to the configuration task is to start with the set of compon on the order and construct an acceptable configuration using its knowledg the constraints on component relationships. Since it's strategy is wholehart bottom-up, it may seem surprising that R1 is able to generate an accept configuration without doing backtracking. R1 is able to avoid search becaus task domain permits the use of the Match method [9]. Match will be discus at length in Section 2.3 below. Basically, the method is applicable if i possible to order a set of decisions in such a way that no decision has to made until the information sufficient for making that decision is available. T applicability condition is trivially satisfied if the set of decisions required b task is fixed and can be ordered a priori. But in the VAX-11/780 configurat task, the set of decisions required varies widely from configuration to configu tion. The Match method is a procedure for dynamically ordering a set of decisi When Match is used, each decision brings one closer to the successful complet of the task.

Although a significant portion of this paper is devoted to a description precisely how R1 goes about doing the configuration task, I have tried to av letting the details of R1's inner workings overshadow the domain independ lessons that have emerged from this research. There are two important lesso

- An expert system can perform a task simply by recognizing what to provided that it is possible to determine locally (i.e., at each step) whet taking some particular action is consistent with acceptable performance on task.
- When an expert system is implemented as a production system, the job refining and extending the system's knowledge is quite easy.

[2]R1's output for a sample order is shown in Appendix B.

The paper is divided into three sections. The first section describes the VAX-11/780 configuration task and characterizes its difficulty. The second section describes R1 and discusses its evolution from a system with only the most limited capabilities to what might fairly be called an expert. The third section describes R1's current level of expertise and isolates the design decisions that made the building of R1 straightforward.

1. The Task

The VAX-11/780 is the first implementation of Digital Equipment Corporation's VAX-11 architecture. It is similar in many respects to the PDP-11, though its virtual address space is 2^{32} rather than 2^{16}. The VAX-11/780 uses a high speed synchronous bus, called the sbi (synchronous backplane interconnect), as its primary interconnect. The central processor, one or two memory control units, one to four massbus interfaces, and one to four unibus interfaces can be connected to the sbi. The massbuses and particularly the unibuses can support a wide variety of peripheral devices. Because the number of system variations is so large, the VAX configuration task is nontrivial.

1.1. The size of the task

A configurer must have two sorts of knowledge. First, he must have information about each of the components that a customer might order. For each component, the configurer must know the properties that are relevant to system configuration—e.g., its voltage, its frequency, how many devices it can support (if it is a controller), how many ports it has; I will call this knowledge *component information.* Second, he must have rules that enable him to associate components to form partial configurations and to associate partial configurations to form a functionally acceptable system configuration. These rules must indicate what components can (or must) be associated and what constraints must be satisfied in order for these associations to be acceptable; I will call this knowledge *constraint knowledge.*

The difficulty of the VAX configuration task is a function of the amount of component information and the amount of constraint knowledge required to perform the task. It is fairly easy to estimate the amount of component information that is needed. On the average, a configurer must know eight properties of a component in order to be able to configure it appropriately. Currently about 420 components are supported for the VAX.[3] Thus there are over 3300 pieces of component information that a VAX configurer must have access to.

Before R1 was developed, it would have been difficult to estimate accurately

[3]Of the 420 components, about 180 are actually bundles composed of various subsets of the remaining 240 components.

this knowledge from them, it became clear that their knowledge takes to
(1) The experts have a sparse but highly reliable picture of their task d
When asked to describe the configuration task, they do so in terms
subtasks involved and the various temporal relationships among these sul
(2) They also have a considerable amount of very detailed knowledg
indicates the features that particular partial configurations and unconf
components must have in order for the partial configurations to be exten
particular ways.Both sorts of knowledge are easily expressable as r
extracted 480 rules. Of these, 96 define situations in which some subtask
be initiated. The other 384 rules define situations in which some
configuration should be extended in some way.

1.2. The constraints

To understand a program, such as R1, that is driven almost entir
task-specific knowledge, it is necessary to understand the nature of th
straints imposed by the task. This subsection provides an example of a s
subtask that can arise within the configuration task and indicates
constraint knowledge involved, (2) the informational demands imposed
constraint knowledge, and (3) the extent to which the subtask presu
other subtasks. The subtask is that of placing unibus modules in backplar
is the case with the configuration task in general, many of the constraints
subtask have a strongly ad hoc flavor; they are not easily derivable fron
general knowledge because they are the result of design decisions
justifications are hidden in a shadowy past. Moreover, the applicabili
particular constraint is ordinarily dependent on a variety of factors;
decisions that a configurer makes must take a large number of deta
account.

Whenever more than one unibus option is ordered for a VAX, it is ne
to place the modules on the unibus in an acceptable sequence. Ignoring
constraints, it is straightforward to determine the theoretically
sequence for the modules; the modules are sorted on the basis o
interrupt priority and within that on the basis of their transfer rate. B
module can be placed on the unibus, it is necessary to select a bac
Several constraints come into play. Backplanes come in two sizes (4-s
9-slot) and can have any of several pinning types. The backplane selecte
be of the pinning type required by the unibus module. To determine the
the backplane to be selected, it is necessary, first, to determine whether
is constrained by the box that the backplane will be placed in. A b
accommodate five 4-slot backplanes; the exception is that a 9-slot bac

may not occupy the space reserved for the second and third 4-slot backplanes.[4] Assuming that either a 4-slot or a 9-slot backplane would be acceptable, the next constraint to come into play is that a 9-slot backplane should not be selected unless the next N modules in the optimal sequence all require a backplane of the same type and will not all fit in a 4-slot backplane. Once a backplane is selected, the board or boards comprising the next module in the optimal sequence can be placed in the backplane. However, the first and last slots of a backplane cannot accomodate a full width board; thus the configurer must make sure that the boards will physically fit into the backplane. There are several other constraints. For example, the total amount of power that can be drawn from a regulator is limited; also, if the length of the unibus exceeds a certain limit or if the load on the unibus exceeds a certain limit, a backplane containing a unibus repeater must be placed in the box.

After a module has been placed in a backplane, there is frequently room for additional modules, but the next module in the optimal sequence may require a backplane of a different type or more space than is available in the backplane. At that point the configurer must decide whether to deviate from the optimal sequence or to leave some of the backplane slots empty; the decision is based on the total amount of box space available and the seriousness of the deviation. If there is sufficient box space to accommodate the modules when they are in the optimal sequence, the optimal sequence should be preserved (even if this entails adding additional backplanes to the order). If there is not sufficient space, the seriousness of the deviation must be determined; there are some less than optimal sequences that are acceptable. If the decision is that the deviation from the optimal would impair the functionality of the system, then the configurer must add another box (and possibly a cabinet as well) to the order; if the decision is that an acceptable suboptimal sequence can be found, then some module other than the next one in the optimal sequence is placed in the backplane. In general, it is acceptable to add a module that is not next in the optimal sequence if it meets all of the constraints mentioned above and has a transfer rate that is lower than that of any other module with the same interrupt priority that it will precede.

There are reasons, other than lack of slot space or power, why the configurer must consider deviating from the optimal sequence. If the module that is to be added is a multiplexer, for example, then in addition to the space and power constraints, there is the added constraint that sufficient panel space must be available in the cabinet containing the box that will contain the multiplexer. If there is not enough panel space in that cabinet, the configurer must decide whether to deviate from the optimal ordering or to put all of the remaining modules in the remaining cabinets. Again, the decision must be made on the

[4]The box that contains unibus modules has two +5 volt regulators. One of these regulators supplies power to the first two slot backplanes (or to the first 9-slot backplane); the second supplies power to the other backplanes. All of the modules in a backplane must draw power from the same regulator.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.