AI Magazine Volume 5 Number 3 (1984) (© AAAI)

R1 Revisited: Four Years in the Trenches

Judith Bachant

Intelligent Systems Technology Group
Digital Equipment Corporation
Hudson, Massachusetts 01749

John McDermott

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

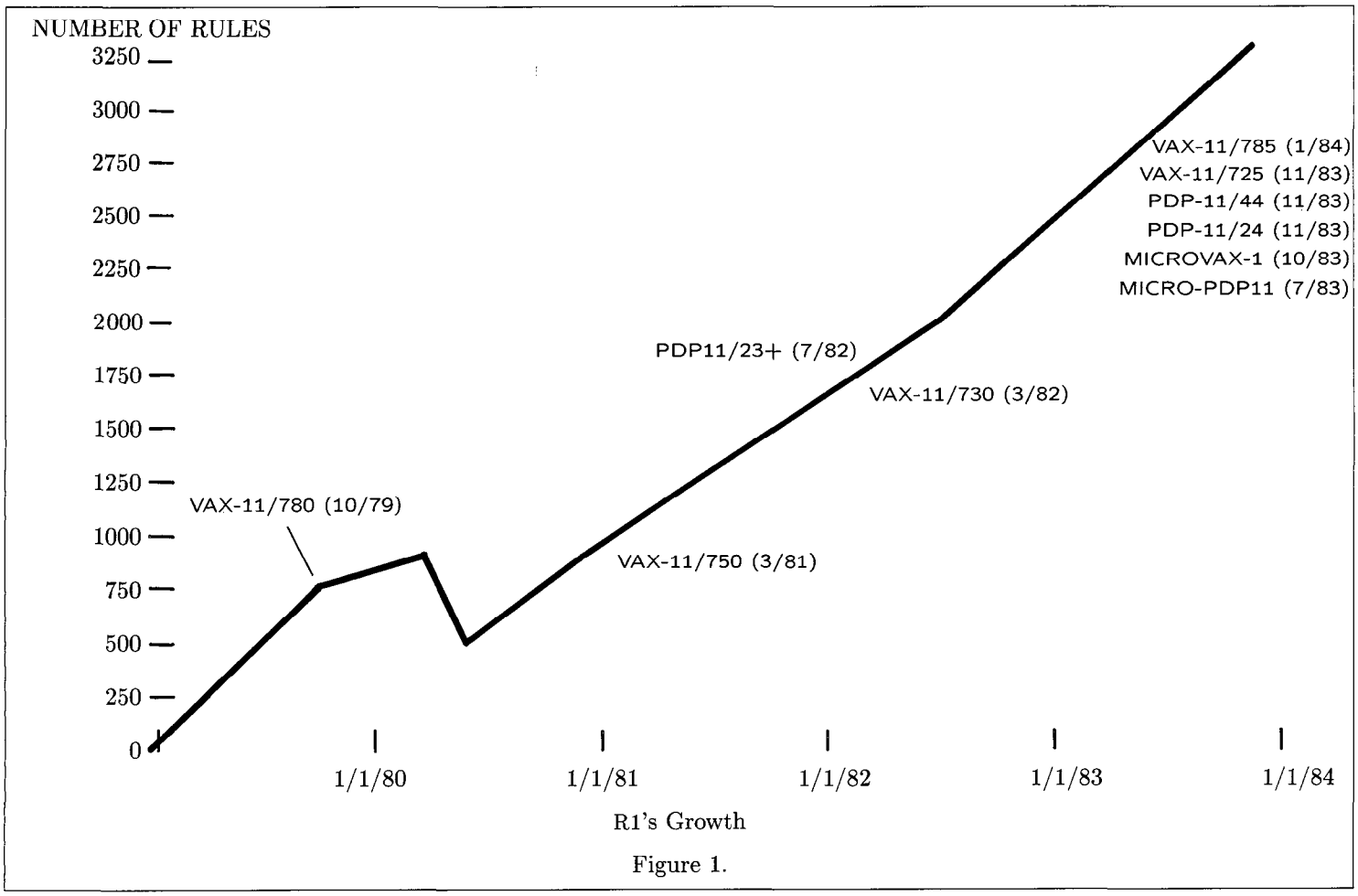
In 1980, Digital Equipment Corporation began to use a rule-based system called R1 by some and XCON by others to configure VAX-11 computer systems In the intervening years, R1's knowledge has increased substantially and its usefulness to Digital continues to grow. This article describes what is involved in extending R1's knowledge base and evaluates R1's performance during the four year period.

IN THE SUMMER 1981 ISSUE of the AI Magazine, an article entitled "R1: the formative years" described how a rule-based configurer of computer systems had been developed and put to work (McDermott, 1981). At the time that article was written, R1 had been used for only a little over a year and no one had much perspective on its use or usefulness. R1 has now been configuring computer systems for over four years. This experience has provided some insight into the ease and difficulty of continuing to grow an expert system in a production environment and into the kind of performance expectations it might be reasonable to have about a current generation rule-based system.

The approach R1 takes to the configuration task and the way its knowledge is represented have been described elsewhere (McDermott,1980) and (McDermott, 1982). Briefly, given a customer's purchase order, R1 determines what, if any, substitutions and additions have to be made to the order to make it consistent, complete, and produce a number of diagrams showing the spatial and logical relationships among the 50 to 150 components that typically constitute a system. The program has been used on a regular basis by Digital Equipment Corporation's manufacturing organization since January, 1980. R1 has sufficient knowledge of the configuration domain and of the peculiarities of the various configuration constraints that at each step in a configuration task it is usually able to recognize just what to do; thus it ordinarily does not need to backtrack when configuring a computer system.

At the beginning of R1's development, no clear expectations existed about how long it would take to collect enough knowledge to make R1 an expert. We did expect that at some point the rate at which R1 would acquire new knowledge would at least slow, if not stop. We even thought that R1 would be done eventually (that is, R1 would enter a maintenance mode of well-defined and minor additions, interspersed with occasional bug fixes.) It is difficult now to believe R1 will ever be done; we expect it to continue to grow and evolve for as long as there is a configuration task. It may be that if R1's domain were less volatile, R1 would not require perpetual development. But it is

A large number of people have played critical roles in R1's development. Among those who deserve special mention are John Barnwell, Dick Caruso, Ken Gilbert, Keith Jensen, Allan Kent, Dave Kiernan, Arnold Kraft, Dennis O'Connor, and Ed Orciuch. We want to thank Allen Newell, Dennis O'Connor, and Ed Orciuch for their helpful comments on earlier drafts of this article



R1's Growth

Figure 1.

probably also true that if the domain were less volatile, the task would not require a knowledge-based system.

The early expectations about R1's performance were likewise vague, except just as R1 was beginning to be used, a Digital employee responsible for the configuration process predicted that for R1 to be useful, 90% to 95% of its configurations would have to be perfectly correct. This performance goal is interesting, not so much because R1 took three years to reach it, but because it turned out to be completely wrong. R1's task is just one small part of a process designed to ensure that high quality computer systems are built. Significant redundancy exists in the process precisely because historically no individual has both known enough about configuration and been able to pay close enough attention to each order to be entrusted with the total responsibility. R1 was able to provide significant assistance even when it knew relatively little because the people who used R1 did not demand more of it than of its human predecessors. The one definite performance expectation almost everyone had about R1 in its early days was that it would always configure the same set of components in the same way. It is obvious now and should have been obvious then that this expectation could have been satisfied only if R1 had been discouraged from becoming more expert.

These expectations about R1's developmental and performance histories introduce the two parts of the article. In the next section, the focus will be on the kind of involvement required to extend R1's knowledge base. The final section's focus will be on the kinds of erroneous behavior R1 has exhibited.

R1's Developmental History

This section provides a somewhat anecdotal trip through R1's past. Although it mentions the first year, when most of the activity was at Carnegie-Mellon University [CMU], the primary focus is on the four following years, after R1 began to be used at Digital. When CMU handed over the initial version of R1 to Digital in January 1980, Digital scrambled to put an organization in place that could continue its development. This organization, currently known as the Intelligent Systems Technologies group, began with only five individuals, none of whom had any background in AI. Over the past four years, the group has grown to 77 people responsible for eight different knowledge-based systems, one of which is R1. As R1 was developed, an attempt was made to effect a division of labor between those people responsible for representing R1's knowledge and those responsible for collecting and validating that knowledge. Of the initial technical people, one was an engineer who played the roles of both

a domain expert and of an interface to other domain experts outside the group; the other three people took the knowledge collected by the engineer and formulated it so it was compatible with R1's other knowledge. When the organization was a little over two years old the technical group had grown to eight people, five of whom were responsible for encoding the knowledge collected and validated by the other three. The size of the R1 technical group is still about eight. Now, however, less of a distinction exists between the people responsible for knowledge encoding and those responsible for knowledge collection.

The Knowledge R1 Acquired

Over the past four years, the amount of effort devoted to adding knowledge to R1 has remained relatively constant at about four worker-years per year. And R1's knowledge has grown at a relatively constant rate, though the focus has shifted around. At times the task of eliminating inadequacies in R1's configuration knowledge has received the most attention; at other times, the group's energies have been directed primarily at broadening R1's abilities in various ways. Figure 1 shows the rate at which R1's knowledge has grown; the points in time at which R1 became able to configure new system types are marked. Figure 1 does not show the amount of product information to which R1 has access. This information, which is stored in a data base, is a critical part of the body of information needed to configure a computer system correctly. R1 retrieves the description of each component ordered before it begins configuring a system; while configuring the system, if it determines some piece of required functionality is missing, it searches the data base for components that will provide that functionality. R1 currently has access to almost 5500 component descriptions. We do not have good data on the rate at which the data base has grown, but what data we have suggest the growth rate is quite irregular.

In this article, R1's growth is measured in number of rules. The following values hint at the amount of knowledge an R1 rule contains. The average conditional part of one of R1's rules has 6.1 elements (the minimum number is 1 and the maximum 17). Each element is a pattern that can be instantiated by an object defined by as many as 150 attributes. On the average, a pattern will mention 4.7 of those attributes (the minimum is 1 and the maximum 11) and restrict the values which will satisfy the pattern in various ways. The tests are mostly simple binary functions that determine whether some value in the object has the specified relationship to some constant or to some other value in that or another object. The action part of an average rule has 2.9 elements (the minimum is 1 and the maximum 10). Each element either creates a new object or modifies or deletes an existing object. A rule can be applied when all of its condition elements are instantiated.[1]

[1]For additional information about the nature of R1's rules as well as those of other systems written in OPS5, see (Gupta, 1983)

Work on R1 began in December 1978. During the first four months, most of the effort was on developing an initial set of central capabilities. The initial version of R1 was implemented in OPS4, a general-purpose rule-based language (Forgy, 1979). By April, R1 had 250 rules. During the same period, a small amount of effort was devoted to generating descriptions of the most common components supported on the VAX-11/780. After this demonstration version of R1 had been developed, most of the effort during the next six months was divided between refining those initial capabilities and adding component descriptions to the data base; in October 1979, R1 had 750 rules and a data base consisting of 450 component descriptions. During the following six months, little development work was done on R1 either at Digital or CMU because the main focus was on defining a career path for R1 within Digital. But beginning in April 1980, three months were spent at CMU in rewriting the OPS4 version of R1 in OPS5 (Forgy, 1981). Given that the knowledge was already laid out in the OPS4 version, a variety of generalizations emerged and the resulting system, though more capable, had only 500 rules.

By the end of 1980, R1 had 850 rules, most of which were added by people at CMU to provide R1 with additional functionality; the primary focus at Digital during the second half of 1980 was on adding component descriptions to the data base and providing a group of people with the skills to take over the continued development of R1. Most of the work on R1 since early in 1981 has been done by people at Digital. By March 1981, the group at Digital had extended R1 so it could configure VAX-11/750 systems. During the remainder of 1981, most of the group's effort was focused on refining R1's knowledge of how to configure VAX-11/780 and VAX-11/750 systems. In 1982, the focus changed to extending R1 to cover more systems. While some effort was spent in improving R1's performance, substantial effort was spent in extending its scope. By March, a few months before the VAX-11/730 was announced, R1 was able to configure VAX-11/730 systems, and by July, R1 was able to configure PDP-11/23+ systems. At that point, R1's knowledge base consisted of about 2000 rules. The remainder of 1982 and the first few months of 1983 were devoted primarily to refining that knowledge. At that point, a concerted effort was made to extend R1's capabilities so it could configure all the systems sold by Digital in significant volume. When that task was finished in November 1983, R1 had about 3300 rules and its data base contained about 5500 component descriptions. While a significant amount of time will continue to be devoted to extending R1's capabilities to cover new systems as they are announced, effort will also be spent in continuing to deepen R1's expertise in the configuration domain.

As Digital has become more dependent on R1, it has become increasingly important that R1 be highly reliable. Thus substantial attention has been paid to the question of how to combine the demands of reliability with those of continuous development. Early on, little attention was paid to formalizing the developmental process; as problems were reported,

	NUMBER OF RULES	AVERAGE RULES PER SUBTASK	AVERAGE NUMBER OF PARTS ORDERED	AVERAGE RULE FIRINGS	PERCENT OF KNOWLEDGE FREQUENTLY USED	NUMBER OF PARTS IN THE DATABASE
THE INITIAL R1	777	7 6	88	1056	44%	420
THE CURRENT R1	3303	10 3	78	1064	47%	5481
VAX-11/785	2883	9.8	163	2654	24%	3398
VAX-11/780	2883	9 8	171	1925	31%	3398
VAX-11/750	2801	9 7	111	1300	29%	2915
VAX-11/730	2810	9 7	85	1141	29%	2489
VAX-11/725	2788	9 7	34	622	8%	1981
MICROVAX-1	1516	7 3	34	546	18%	1490
MICRO-PDP11	1516	7 3	44	546	18%	1828
PDP-11/23+	1516	7 3	49	608	20%	1894
PDP-11/24	2786	9.7	43	567	13%	1763
PDP-11/44	2786	9 7	43	733	15%	1764

A comparison of the initial and current versions of R1.

Figure 2.

individuals would collect the needed knowledge, add it to the system, and depending on the press of other problems, do more or less testing to determine that the overall capability of the system had not worsened. As time passed, the developmental process acquired substantially more structure. Planned release dates are now preceded by extensive testing of the system.

The article describing the initial version of R1 (McDermott, 1982) provides some insight into the nature of R1's knowledge by presenting a variety of measurements. Figure 2 compares the measurements from the initial version of R1 with corresponding measurements from the current version. Since a significant amount of the knowledge in the current version is specific to just a subset of the system types it can configure, Figure 2 provides the measurements for system-specific configurers as well as for the union of those configurers. Until recently, instead of a single version of R1 that could configure all system types, there was a version of R1 for each system type. Each of these versions consisted of a set of from 50 to 100 rules specific to a system type and two much larger sets of rules; it shared one of these rule sets with all of the other system types and the other with the system types having the same primary bus. About 300 of the shared rules were themselves specific to just one of the system types; each of these rules was included with the shared rules because it was relevant to a shared subtask.

R1's rules are grouped together on the basis of the subtask to which they are relevant; the "number of rules" column displays the total number of rules available to R1 in performing the configuration task, and the "average number of rules per subtask" column displays the mean number of rules in a group. The 3303 rules the current R1 has is the union of the rules of each system-specific configurer; the 10.3 rules per subtask is the union of the groups of rules the system-specific configurers bring to bear on a particular task. The "average number of parts ordered" column displays the number of components R1 has to configure. This number is significantly larger than the number of components listed on a purchase order since those line items actually refer to bundles of configurable components.

The numbers in the "average rule firings" and "percent of knowledge frequently used" columns are based on small sets of runs. For the initial R1, the numbers came from running R1 on 20 typical orders. For the current R1, the numbers came from running each system-specific version of R1 on about 20 orders of comparable complexity. The "average rule firings" column shows that substantially more is done in configuring a VAX-11/780 order now than was done initially; almost twice as many rules are applied. Two factors contribute to this increase. The configuration task has been enlarged by definition (*i.e.* there is now more to do), and second, there has been an increase in the average number of components per order.[2]

The "percent of knowledge frequently used" column shows what percentage of the rules are used at least once in at least one of the sample runs. Thus for the initial R1, 44% of the 777 rules were applied at least once over the 20 sample runs, and for the current R1, 47% of the 3303 rules were applied at least once over the approximately 200 sample runs. The fact that a substantial fraction of R1's knowledge is used only rarely is, of course, just what we would expect of a knowledge-based system. But the percentages for the system-specific versions are somewhat misleading. We would expect the percentage for each version to be lower than the overall percentage because each was run on only about 20 orders. However, because each version has knowledge that is not relevant to its tasks, the percentages for the versions are lower than they otherwise would be. The percentages for the VAX-11/780, the VAX-11/750, and the VAX-11/730 are the most accurate, but even they are too low by several percentage points. Since the nature of the knowledge used by

[2]On the average, 1.67 VAX-11/780 cpu minutes are required to configure an order.

each version is quite similar, it is likely that the percentage of the knowledge frequently used by each is pretty much the same—somewhere between 35% and 40% .

About 65% of the 2526 rules added to R1 since 1980 extend R1's general configuration capabilities; only about 35% of the rules are specific to a single system type. Of the 65% at least 15% were added to correct or refine knowledge of how to perform some subtask. This lower bound is suggested by the fact that the "average number of rules per subtask" increased by 30% during the past four years (*i.e.*, about 230 rules were added to the groups of rules applicable to the subtasks the initial R1 knew how to perform); adding a rule to the group applicable to some subtask is almost invariably done to correct or refine the knowledge of how to perform that subtask. The 15% is a lower bound because as the knowledge required to perform some subtask grows, it may become evident that what was viewed as a single subtask can be viewed as two or more simpler subtasks; what we do not know is how much the average number of rules per subtask would have grown if this subtask splitting had never occurred.

The Kinds of Changes R1 Has Undergone

As it turned out, the task of developing R1 had just begun when it was first put into use. In this section, we attempt to give a flavor of the kinds of changes that have been made to R1 over the past four years by examining a few examples in some detail. Our primary purpose in examining the growth of R1's knowledge is to better understand what is involved in adding knowledge to such a system. We can identify four reasons why knowledge was added to R1:

- To make minor refinements (adding knowledge to improve R1's performance on an existing subtask);
- To make major refinements (adding the knowledge required for R1 to perform a new subtask);
- To extend the definition of the configuration task in significant ways.

Ordinarily when people talk about why knowledge is added to an expert system, they seem to have the first reason in mind. As we have seen, of the more than 2500 rules added to R1 during the past four years, the data in Figure 2 suggest that more than 10% have been added to make minor refinements, fewer than 40% have been added to make major refinements, at least 35% have been added to provide functionality needed to deal with new system types, and perhaps as many as 15% have been added to extend the definition of the task in significant ways.

Minor Refinements. A knowledge addition of the first type is required when R1 cannot perform some subtask that it was thought to be able to perform. For example, over the years R1 has made several errors involving the placement of backplanes in boxes. One instance of such an error has to do with a backplane's location. In one variety of a 24 slot box, because of power considerations, a backplane is not permitted to cover slot 10. R1 knew that if it covered slot 10 when placing a backplane, it needed to move that backplane toward the back of the box so the backplane's front edge would be in slot 11. R1's knowledge was incomplete because it did not know it had to move any previously placed backplane from the front of the box toward the middle so that its back edge would be in slot 9. This backplane has to be moved toward the middle because leaving a larger space between the two backplanes would mean the standard cable used to connect backplanes could not be used (since it is not long enough). Fixing R1 was a straightforward task, but it required a certain amount of creativity (*i.e.*, it was not just a matter of "adding some more domain knowledge.") What R1 lacked was any notion of "deliberately vacant space." In order to provide rules that could recognize situations in which blank space was inappropriately positioned, R1 had to have the concept of blank space and an understanding of how to make a note that a particular space had been left blank on purpose. Given this, it was straightforward to add a few rules that recognized when some piece of blank space was inappropriately located and swap it with a backplane.

Major Refinements. A knowledge addition that results in a major refinement to R1 can be made in two kinds of situations: when R1 does not have any knowledge about how to perform some subtask, and when its knowledge of how to perform some subtask becomes so tangled that ways need to be found of representing the knowledge more generally. Brief examples of both situations are presented below; in the following section we provide a more lengthy analysis of one attempt to rewrite a set of rules, initiated almost purely to increase generality and understandability.

Most of the modules R1 configures on a UNIBUS consist of one or more boards that plug into backplanes which go in boxes. If multiple boards are required, they are usually placed next to each other in the same backplane. A situation unfamiliar to R1 arose when a module was designed with boards on two buses. Its first board was to be configured in an SPC backplane while the three remaining boards were to be configured in a special backplane that had to be located in the same box as the first board, but not in the same backplane. One way of extending R1 to handle this new component would have been to use a look-ahead strategy; R1 would have checked for space, power, and cabling constraints on the special backplane before configuring the first board. An alternative would have been a simple backtracking strategy. The approach R1 actually took involved a combination of both look-ahead and backtracking. R1 applies the same rules it uses for other modules to configure the first board; a few special rules then try to foresee abstract constraint violations involving the rest of the boards. If a problem is found, the first board is unconfigured. If no constraints are violated, power and space are reserved for the remaining boards.

Early in R1's history, only two types of panels needed to be considered. A few rules were sufficient to guard against the possibility of trying to configure two panels in the same

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.