Artificial Intelligence 59 (1993) 197–204
Elsevier

ARTINT 977

Notes on "Epistemology of a rule-based expert system"

William J. Clancey

Institute for Research on Learning, 2550 Hanover Street, Palo Alto, CA 94304, USA

1. Introduction

When I joined the Mycin project in January 1975, I was full of excitement about working with physicians and doing something "relevant to society". This excitement soon paled as I became lost by medical jargon and tedious design arguments. In April 1975, I wrote an essay, "Why is the Tetracycline rule so difficult to understand?" This simple rule presented itself as a puzzle: "If the patient is less than 7 years old, then do not prescribe Tetracycline". I knew what all the words meant, but I couldn't understand why the rule was correct. How could Mycin understand the rule? What did it mean to understand a rule? And if Mycin didn't understand the rule, how could it be said to be reasoning? Somewhat fatuously, I labeled my line-printer listing of rules "What Mycin Knows". More than a decade would pass before I realized that to have a representation in your pocket is not to be intelligent, that "having knowledge" does not mean possessing some *things*. A more accurate label for the rule listing would have been "Our representations of what Drs. Shortliffe, Blum, and Yu know".

But criticizing Mycin wasn't on my mind in 1975, before we even called Mycin an "expert system", before even the Heuristic Programming Project became the Knowledge Systems Lab. My interest was to understand the medicine and logic behind the rules. Within the AI research goals of the time, this was framed as research on explanation and teaching (Scott et al.

Correspondence to: W.J. Clancey, Institute for Research on Learning, 2550 Hanover Street, Palo Alto, CA 94304, USA. E-mail: clancey@xerox.parc.

[21], Clancey [4]). The "epistemology" article [5] was originally a chapter of my dissertation [7].

By late 1978, my interest in ordering and understanding Mycin's rules focused on the rules we (principally Victor Yu and Larry Fagan) had recently added for treating meningitis. By this time, Jim Bennett and I were influenced by Randy Davis' work on knowledge acquisition [13], and intrigued by ways in which the process could be managed and taught. From experience, we knew that building a knowledge base wasn't a matter of just generating rules and dropping them into a pot. Jim's experience in constructing Sacon [1] showed the importance of helping experts write useful rules, in particular, organizing rules conceived in isolation into lines of reasoning.

2. Structure–strategy–support design rationales

In effect, we were already moving from the programmer's view of rules as modular and independently changeable to the view that a knowledge base was designed and maintained as a coherent whole. The Mycin team was impressed by the effort in 1975–77 required to generate and test different versions of Mycin's context tree. We were attempting to represent reasoning about multiple cultures from a given body site at different points in time. Clearly, the contribution by the computer scientists was not just in developing the inference engine and attendant subsystems, but in formulating and encoding a complex reasoning network. Indeed, in formulating new concepts and causal models, we were involved in medical science.

By 1978, Bennett and I focused on the idea of *inference structure*, an outline of the dominant goal structure of a rule set. Seeing the goal structures of Mycin and Sacon on paper, we wondered, "What is the logic behind these trees?" Here I was strongly influenced by Brown, Collins, and Harris' paper, "AI and learning strategies" [3], which analyzed reasoning strategies in different domains. I noted the parallel between controlling Mycin's rules and deciding which axiom to apply in algebraic simplification [5, Fig. 12]. Until this time, the idea of strategy in the Mycin project was simply a rule that controlled other rules (a *metarule*). But here I realized that a strategy was a kind of argument. It had its own logical content, which for medical diagnosis was more like a *procedure* for focusing an extended dialogue, than isolated metarules affecting rule ordering. This was probably the most important idea in the design of Neomycin, which was already under way by the fall of 1979.

The procedural–declarative controversy also influenced my analysis. What was the relation between "explicit" and "implicit knowledge"? In what sense does clause ordering constitute implicit knowledge? For example, I can pro-

ceduralize the relation between "age of the patient" and "alcoholism" by placing the age clause before the alcoholism clause in rules. But how is this different from the "declarative representation", "If the age is less than 13, then the patient is probably not an alcoholic"? Since Mycin doesn't understand what alcoholism is anyway, in what sense is the rule's meaning more explicit? It was many years before I realized that there was no such thing as "the meaning" of a representation. And we were light years from distinguishing between "knowing a meaning" and "a representation of meaning". At the very least, the ideas of implicit and explicit knowledge encouraged me to list and study relations such as clause ordering that were crucial to the program's correct operation. Ultimately, I framed this analysis by relating it to Woods' "What's in a link?" [26] and Brachman's "What's in a concept?" [2]. Today, with more respect for simplicity, I would have called my article, "What's in a rule?".

Looking back, I believe that the "strategy–structure–support" framework holds up. The observation that an inference procedure (strategy) indexes the domain model (the knowledge base of propositions and rules) through a vocabulary of relations (structure, the domain theory) is of basic importance for understanding how changes to the theory, model, and inference procedure interrelate. Unfortunately, my misleading characterization of strategic knowledge as domain-independent (instead of domain-general) led some people to belittle the idea of representing the domain model separately from the inference and communication procedures. I believe these distinctions are central for articulating and advancing the process representation techniques of AI programming [11].

3. Knowledge bases and generative models

After constructing Neomycin and studying its metarules in the mid-1980s, I realized that removing ordering relations between rule clauses produced a special kind of model, like the relation between a grammar and a lexicon. In effect, we discover patterns in expressions (Mycin's rules) and formulate a grammar that can generate these patterns by interpreting a separate set of propositions (the domain model of propositions and rules). I believe that understanding how such process models are constructed is important for understanding the capabilities and limits of qualitative modeling.

The first idea is that general theories of processes are developed by abstracting *ordering patterns* (e.g, clause ordering, rule ordering, data-request ordering) from a collection of models written in some language [8]. We followed this approach in creating Neomycin from Mycin, and then in studying patterns in Neomycin's metarules [9]. A more general observation is that *different procedures* for interpreting a domain model (e.g., an infer-

ence procedure, a compiler, an explanation program, a student modeling program) index the model through different classifications of domain terms and relations [11].

Second, a domain-general inference procedure (general because it uses variables for domain terms) can be used like a natural language grammar to parse an expert or student's sequence of requests for data and hypothesis statements (Wilkins et al. [25]). A given sequence of expert or student behavior can be parsed in different ways, depending on different assumptions about the person's domain model [9].

Third, when we replace a representation by what appears to be its generative constituents, we specialize it (by specifying when propositions and rules should be applied), and potentially lose robustness. For example, if we replace the dictum "generalize questions" (formalized in Neomycin by a metarule) by the conditions under which this should be done (e.g., you are in a hurry and this finding is rarely present), we will require both more details to be represented (e.g., the frequency of a finding) and more information to be checked during the problem-solving context (e.g., am I in a hurry today?). That is, by making more aspects of the system being modeled and the context explicit, more reasoning (or search of triggering conditions) is required. Besides requiring more data, the program will no longer apply the rule in situations in which it might be relevant. Recent research in self-organizing and reactive architectures is partly motivated by these observations (Steels [22]).

Of course, these statements would not have been made in the 1970s, when few people acknowledged that Mycin's classification of diseases and causal relations expressed in rules constituted a model of the domain. We needed to recognize that we were constructing new models in knowledge bases before we could understand how people typically create and use models, and the capabilities of a reasoning mechanism built only out of models.

4. The role of rationalization reconsidered

We originally conceived of support knowledge as the causal and social context that justifies a rule, an objective documentation for why a rule is correct. Today we would call the justification a *design rationale*, and emphasize its use in developing better models over time. In particular, any number of rationalizations for or against a representation are possible, depending on changing circumstances in which representations are interpreted. Rather than viewing justifications as objective and pertaining to "truth", a design rationale should cite the shortcomings in the previous theory or model, relative to how it has been used (Schön [20]). Rationalization of this

sort involves reconceiving the goals of the program and what constitutes a problem, not just providing "deeper" explanations for isolated rules.

Conflating knowledge and knowledge representations distorts how people create models, and leads to an inadequate conception of what tools might help us. For example, in the "epistemology" article [5] I observed that the meaning of represented concepts tends to be generalized when later additions are made to the knowledge base, which I termed *concept broadening.* I now believe that this is an inherent aspect of representational change [10], not something that can be prevented by "principled representations". Tools for maintaining and using design rationales, grounded in a case library, could take this into account.

Our conception of explanation systems was also biased by the view that knowledge bases weren't our models, but were delivered from the heads of experts. We didn't consider that users were already representing knowledge in their everyday affairs. We believed that a program's explanation would help experts improve the program (transmission from expert to program), but in the workplace we conceived of explanation as a teaching device (transmission from program to user). We viewed changes to the rules in terms of debugging a program, expecting it to be a convergent process that would be controlled by technicians, or preferably the experts themselves. After all, building a knowledge base was conceived as a process of acquiring already "known" knowledge from experts. To the extent users had knowledge, it was assumed to be unarticulated, and hence incomplete and naive.

We viewed all interactions between people and program in terms of "transfer": Experts had knowledge stored in their heads. The knowledge base was an objective inventory of expert knowledge. Users lacked knowledge. The role of consultation (and knowledge acquisition and teaching) was to transfer knowledge between experts, users, and students. Knowledge engineers are like priests; they receive "The Word" from experts above, add nothing to the content, but codify it accurately into written rules, and pass it down to ordinary folks as commandments to live by. If you are not an expert, you learn by being told. This is how knowledge engineers learn from experts and how users and students learn from the expert system. This view lacks any sense that the knowledge base could belong to a community of practitioners, the users and experts alike, and might be developed and maintained by them (but see Stefik and Conway [23]).

Similarly, apart from courtesies of the bedside manner, it was difficult to conceive what a medical student should be taught, other than Mycin's rules. We didn't realize that the complement of Neomycin—how the model relates to the unformalized world of medical practice—is an essential part of what a student needs to know. What explanations could help students and consultation users become more aware of the quality of their work, know when to question what they are doing, and generate ideas for changing what

they are doing? In other words, how can we use knowledge representations to make people *reflective practitioners* [20]?

As an example, consider the disease taxonomy of Neomycin. Today we view such representations not as a *product* to be delivered to a student, but as a partial model of a practice. Besides learning the various diseases and their relations, we would want the student to learn the following:

- Why do we taxonomize diseases?
- What gets glossed? How do we know when this stereotypic view of physiological processes is misleading?
- Who knows this taxonomy; what is its origin?
- What is the nature of disagreements; how do they arise; how are they settled?
- How are taxonomies related to medical research? Are they becoming unnecessary as we develop better mechanistic models?
- What are good ways to keep a taxonomic perspective up-to-date?

By this view, knowledge is *knowing how to live* in a community, not just static facts or procedures that *represent* what people do. This is what we want the student to learn and what our computer tools could support (Lave and Wenger [17], Schön [20], Clancey [12], Greenbaum and Kyng [14]). Today we realize that "glass box design" is not an inherent property of a representational system, but a *relation* between the design and the practices of a community (Wenger [24]).

If I had this perspective in 1975, before building Guidon I would have experimented with the original Mycin program in the medical school. I would first give Mycin to the students with illustrative cases demonstrating the program's capabilities. I would then challenge them to defeat the program by selecting new cases from the clinic and explaining why Mycin fails. This would teach them something about computer models as well as the domain.

5. Subsequent research

This article helped frame explanation research in terms of content, as opposed to the goal-rule backward-chaining syntax. But given my view that the program was an omniscient expert, it is no surprise that I didn't discuss the *explanation process* as a kind of negotiation, a joint construction (cf. Moore and Swartout [18]). For example, the Pollack et al. study [19] of radio talk shows reveals how participants engage in problem framing, co-define the expert's capabilities, and evaluate the usefulness of his advice. Related work by Langlotz and Shortliffe [16] explores the relation of consultation dialogs to human involvement and responsibility; Karp and Wilkins [15]

reconsider the nature of rule-based models in relation to explicit causal representations.

This article was a step in the design of Neomycin and my subsequent study of other expert systems. I was surprised to find after completing the paper "Heuristic classification" [6] that the abstraction–heuristic-association–refinement horseshoe diagram is anticipated by Fig. 8, which shows how a rule can be explained by generalizing it. In effect, I have written a trilogy of papers with expanding interests and claims about representation: The "epistemology" article [5] is a study of Mycin's rules; "Heuristic classification" [6] is a generalization to other expert systems; and "Model construction operators" [11] is a further generalization to the process modeling methodology of AI programming.

I am especially grateful to Bruce Buchanan, John Seely Brown, and Danny Bobrow for their persistent interest and direction during more than two years of rewriting this paper.

References

[1] J. Bennett, L. Creary, R. Englemore and R. Melosh, SACON: a knowledge-based consultant for structural analysis, in: *Proceedings IJCAI-79*, Tokyo (1979) 47–49.

[2] R.J. Brachman, What's in a concept: structural foundations for semantic networks, *Int. J. of Man–Mach. Stud.* **9** (1977) 127—152.

[3] J.S. Brown, A. Collins and G. Harris, Artificial intelligence and learning strategies, in: H. O'Neill, ed., *Learning Strategies* (Academic Press, New York, 1977).

[4] W.J. Clancey, An antibiotic therapy selector which provides for explanations, in: *Proceedings IJCAI-77*, Cambridge, MA (1977); expanded version in: B.G. Buchanan and E.H. Shortliffe, eds., *Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project* (Addison-Wesley, Reading, MA, 1984) 133–146.

[5] W.J. Clancey, The epistemology of a rule-based expert system—a framework for explanation, *Artif. Intell.* **20** (3) (1983) 215–251.

[6] W.J. Clancey, Heuristic classification, *Artif. Intell.* **27** (1985) 289–350.

[7] W.J. Clancey, *Knowledge-Based Tutoring: The GUIDON Program* (MIT Press, Cambridge, MA, 1987).

[8] W.J. Clancey, From Guidon to Neomycin and Heracles in twenty short lessons: ONR final report, 1979–1985, in: A. van Lamsweerde, ed., *Current Issues in Expert Systems* (Academic Press, London, 1987) 79–123; also: *AI Mag.* **7** (3) (1986) 40–60.

[9] W.J. Clancey, Acquiring, representing, and evaluating a competence model of diagnosis, in: M.T.H. Chi, R. Glaser and M.J. Farr, eds., *The Nature of Expertise* (Lawrence Erlbaum, Hillsdale, NJ, 1988) 343–418.

[10] W.J. Clancey, Book Review of Rosenfield's *The Invention of Memory: A New View to the Brain, Artif. Intell.* **50** (2) (1991) 241–284.

[11] W.J. Clancey, Model construction operators, *Artif. Intell.* **53** (1) (1992) 1–115.

[12] W.J. Clancey, The knowledge level reconsidered: modeling socio-technical systems, in: K. Ford, ed., Special Issue on Knowledge Acquisition, *Int. J. Intell. Syst.* (to appear).

[13] R. Davis and D.B. Lenat, *Knowledge-Based Systems in Artificial Intelligence* (McGraw-Hill, New York, 1982).

[14] J. Greenbaum and M. Kyng, *Design at Work: Cooperative Design of Computer Systems* (Lawrence Erlbaum, Hillsdale, NJ, 1991).

[15] P.D. Karp and D.C. Wilkins, An analysis of the distinction between deep and shallow expert systems, *Int. J. Expert Syst.* **2** (1) (1989) 1–32.

[16] C.P. Langlotz and E.H. Shortliffe, Adapting a consultation system to critique user plans, *Int. J. Man-Mach. Stud.* **19** (5) (1983) 479–496.
[17] J. Lave and E. Wenger, *Situated Learning: Legitimate Peripheral Participation* (Cambridge University Press, Cambridge, England, 1991).
[18] J.D. Moore and W.R. Swartout, A reactive approach to explanation, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 1504–1510.
[19] M.E. Pollack, J. Hirsherg and B.L. Webber, User participation in the reasoning processes of expert systems, in: *Proceedings AAAI-82*, Pittsburgh, PA (1982) 358–361.
[20] D.A. Schön, *Educating the Reflective Practitioner* (Jossey-Bass, San Francisco, CA, 1987).
[21] A.C. Scott, W.J. Clancey, R. Davis and E.H. Shortliffe, Explanation capabilities of knowledge-based production systems, *Am. J. Comput. Linguistics* (1977) Microfiche 62; also in: B.G. Buchanan and E.H. Shortliffe, eds., *Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project* (Addison-Wesley, Reading, MA, 1984) 338–362.
[22] L. Steels, Cooperation through self-organisation, in: Y. De Mazeau and J.P. Muller, eds., *Multi-Agent Systems* (North-Holland, Amsterdam, 1989).
[23] M. Stefik and L. Conway, Towards the principled engineering of knowledge, in: R. Engelmore, ed., *Readings from the AI Magazine, Volumes 1–5, 1980–85* (AAAI Press, Menlo Park, CA, 1988) 135–147.
[24] E. Wenger, Toward a theory of cultural transparency: elements of a social discourse of the visible and the invisible, Ph.D. Dissertation, Information and Computer Science, University of California, Irvine, CA (1990).
[25] D.E. Wilkins, W.J. Clancey and B.G. Buchanan, On using and evaluating differential modeling in intelligent tutoring and apprentice learning systems, in: J. Psotka, D. Massey and S. Mutter, eds., *Intelligent Tutoring Systems: Lessons Learned* (Lawrence Erlbaum, Hillsdale, NJ, 1988).
[26] W.A. Woods, What's in a link: foundations for semantic networks, in: D.G. Bobrow and A. Collins, eds., *Representation and Understanding* (Academic Press, New York, 1975) 35–82.