Article

The logic of domains

David Ribes (iD)
Human Centered Design & Engineering (HCDE), University of Washington, USA

Andrew S Hoffman
Human Centered Design & Engineering (HCDE), University of Washington, USA

Steven C Slota
Department of Informatics, University of California, Irvine, USA

Geoffrey C Bowker
Department of Informatics, University of California, Irvine, USA

Abstract

The logic of domains has become a key organizing principle for contemporary computing projects and in broader science policy. The logic parses collectives of expertise into 'domains' that are to be studied or engaged in order to inform computational advancements and/or interventions on the domains themselves. The concept of a domain is set against a proposition that there is a more general, domain independent or agnostic technique that can serve to intermediate the domains. This article contrasts instances of this discourse, organizing and techne, drawing from cases in artificial intelligence, software engineering, and science policy to illustrate three ongoing figurations of the logic as i) experimental research, ii) formalization in method and software tools, and iii) a *de facto* organizing principle for science policy and technology development.

Keywords

artificial intelligence, computer science, data science, domains, knowledge representation, science policy

Correspondence to:
David Ribes, Human Centered Design & Engineering (HCDE), University of Washington, 3960 Benson Lane NE, Seattle, WA 98115, USA.
Email: dribes@uw.edu

Introduction

In contemporary settings of applied computational research, such as data science, the use of the term 'domain' is ubiquitous. The term serves to identify, demarcate and characterize spheres of worldly action or knowledge, for instance, biology as the 'domain science' of life or geologists as the 'domain experts' of the earth. The use of the term implies, necessarily, that there is more than one domain, that they are in some way distinct from each other, and thus that domains are topically specific. The concept of a domain is set against a proposition that there is a more general, even universal, method or technique; so, for instance, a data analytic tool may be dubbed 'domain independent', meaning that it can be of use across many, and sometimes all, domains. This general or universal quality is characterized as the feature of a kind of field or expert, recently data science, but in the past a feature of other specializations such as the computing and information sciences.

Our goal is to articulate the regular and changing features of what we call the *logic of domains*. Our strategy of exposition is to contrast three characteristic figurations for the logic, as i) experimental or exploratory investigations of domains, ii) routinization and formalization in toolsets, and iii) naturalization in language, science policy and organizational planning. The first figuration explores the nexus of artificial intelligence and cognitive science that flourished in the 1960s and 1970s as researchers proposed 'to use all of our knowledge to construct a computer program that knows' (McCarthy and Hayes, 1981 [1969]: 6).[1] The second figuration focuses on software engineering, where the logic of domains serves as a pathway for the reuse of software within and across domains. We track the endeavor to formalize inquiry and representation through efforts to transform domain analysis 'from art to engineering' (Arango, 1989: 152). Even while continuing on a trajectory of mounting academic interest in the form of research and tool building, beginning in the 1990s the logic of domains manifests as a novel figure, becoming an organizing principle for science policy and in technology development projects. Herein, domain becomes a vernacular term, and the logic a *de facto* organizational approach, serving to support and fund new research and tool building endeavors as an 'engine of progress' (National Research Council (NRC), 1992) that promises to advance both computational capacity and the domains. As an 'engine', funding is made available for research projects in ways organized from inception by the logic of domains, thus further propelling the logic by supporting new research and tool-building endeavors.

The logic of domains is littered with its own criticisms even amongst its adherents, and throughout the article we recount fragments of key debates as revealing of recurrent challenges for the logic. What is a domain and what constitutes independence varies across these three figurations, even while still exhibiting both positions. In the discussion we offer a synthetic view of the problems that have been articulated by supporters and detractors of the logic of domains, parsing four recurrent topics: ontology; inquiry and representation; accountability; and the relationship of the general and the specific. The conclusion is cast programmatically, calling for research that inspects the practice and consequences of the logic of domains.

Our approach to the investigation of a 'logic' are in part inspired by Goody's (1986) *The Logic of Writing and the Organization of Society*, with the adoption of writing

demonstrating regularities even across distinct regional manifestations. We also draw on Verran's (2001) inspection of the practical creation of equivalencies across distinct formal systems. But we are perhaps most inspired by Mol's (2008) characterization of logic in her study of medical care:

> I am after the rationality, or rather the rationale, of the practices I am studying. Here the term 'logic' helps. It asks for something that one might also call a style. It invites the exploration of what it is appropriate or logical to do in some site or situation, and what is not. It seeks a local, fragile and yet pertinent coherence. This coherence is not necessarily obvious to the people involved. It need not even be verbally available to them. It may be implicit: embedded in practices, buildings, habits and machines. And yet, if we want to talk about it, we need to translate a logic into language. This, then, is what I am after. (p. 8)

And so too are we. What we trace in this article is a genealogy with many influential but untended branches, of direct connections through academic mentorship, shared intellectual milieus, influential writings, concrete passing of technical artifacts, methodologies and standardized software support packages.

The logic of domains is not only a matter of science or research, it has also always been bound up with engineering and design goals, particularly of computational systems, tools or infrastructures that are developed, in part, with the goal of intervening on the domains. Thus, the logic of domains may be characterized as a *style for organizing engineering* (see also Fujimura and Chou, 1994; Hacking, 1992; Keating and Cambrosio, 2012).

Last, our naming for the logic of domains is a light pun, for one recurrent consideration we track is whether domains may have logics of their own, and if so, would that conclude the project of the logic of domains, or offer its very grist?

Domains and their *tertium quid*

The logic parses the world into domains of human action or expertise, along with something beyond or between the domains that is somehow more general, even universal, that serves to intermediate the domains. Throughout the article our key analytic stance is to simultaneously keep in view both domains and independence, even while both positions exhibit changing, if still related, meanings. Much historical and sociological research that has explored the intersection of computing and some domain has failed to recognize intellectual ambitions and technological architectures that seek to transcend domain specificity. It is this goal, to develop cross-domain or domain independent tools, formalisms and methods that is our central focus here, and a key characterizing feature of the logic of domains.

The article is told via three figurations of the logic, roughly arranged in a chronological manner and that share a genealogical relationship across them, but in no sense do we here attempt an historical account of the logic of domains. All three of these figurations can be observed in the present, none are in solely in the past: the logic of domains today remains a research endeavor, a tool-building enterprise, and an organizing principle for science funding and technology development.

What is meant by 'domain' – more precisely, what domains 'have' or 'are made of' – has several different evolving or competing meanings and associated techniques of investigation and representation. In our first figuration, focused on artificial intelligence researchers during the 1960s and 70s, the concept of a domain serves to objectivize the knowledge of circumscribed groups in order to 'capture' and 'encode' it within expert systems. In our second figure, focused on domain analysis in software engineering, domains are the organizational settings that 'have' software tools to solve problems. 'Domain analysis' names an approach that seeks to systematize or formalize inquiry and representation of domains. The third figuration is focused on science policy and a more general vernacularization of the logic of domains. Here, the term domain is regularly deployed but rarely defined, but one common feature is the proposition that *all boats will rise*, that is, that both domain and the computational field will profit from a common endeavor.

The term domain appears consistently even while varying in its referent, but what is beyond, above, between or below domains has a more nebulous designation, a *tertium quid* to the domains: a third position that is indefinite and undefined but is related to two definite or known things. Always cast in relation to domain specificity, it is given such names as 'domain independent', 'domain general', or 'domain agnostic'. On other occasions that position goes unnamed or unremarked, instead referenced by the medium of representation – e.g. 'formalism' or 'knowledge representation language' – used to capture or represent domains.

Paraphrasing Daston and Galison (1992: 82), independence is related to domains as wax to seal, as hollow imprint to the bolder and more solid features of domains. The *tertium quid* to the domains is thus described as independent, general, agnostic, or empty in the sense that it is defined by an absence of dependence, specificity, commitments or content. Yet even while negatively defined against this or that property of a domain, independence is not merely its opposite. The 'stronger test of domain independence' (Van Melle, 1980: 6) is in its additional capacity to be relevant, or applicable, to another domain, many of them, or all of them. In this article, we have included three visualizations that demonstrate the particular image of objectivity established by the logic of domains; each draws on distinct visual conventions, but all posit domain-specific and domain-independent positions in designing a computational system.

What counts as a domain, or independence from domains, shifts across our three figurations. Both domains and independence are the object of contestation, sometimes explicitly while at other times more quietly; their scope and limits are at stake in the very debates we track in this article. We (and occasionally the actors) cast the logic of domains as occurring in a rough spectrum that, on the one end, treats the domains as decomposable in the same way, or, on the other end, treats them as fundamentally distinct while still available to intermediation. Respectively we call these representationalist and irreductionist approaches (Latour, 1988; Stengers, 2000), and return to the topic more extensively in the discussion.

Making the crossing from independence to domains, or vice versa, is a central concern for those adopting the logic. Domain independent tools, techniques, algorithms, or theory must be 'applied', 'tailored', or 'customized' to a specific domain. Alternately,

domain specific features (such as knowledge, software or data) are 'solicited', 'extracted' or 'captured' so as to be rendered independent or transportable within and across domains. We also draw attention to longstanding boundary-spanning figures – variously called the craftsman, domain analyst, knowledge engineer, pi-shaped person, or bilingual – who are simultaneously domain experts *and* capacitated to work beyond the domain. Such boundary-spanning figures proffer a solution to a central goal for those taking up the logic: that software, theory, data or algorithms will be adequate for use in one domain, but also in another, many and, on occasion, all of them.

As the logic of domains has become a vernacular feature of key science policy and technological development projects, it has become a *de facto* style of organizing for tool and system building. Here we limit our focus to policy documents and funding arrangements from the US National Science Foundation and National Academy of Science, but similar statements are apparent in other US science funding agencies and internationally. As a vernacular style of organizing, the logic retains its parsing of the domains, models of empirical inquiry, and its engineering goal to intermediate the domains, but it loses much of the debated intellectual quality that characterized its explicit discussion as an experimental endeavor. In short, beginning in the 1990s, in some spheres 'domain' has become less of an epistemic object and more a matter of fact (Latour and Woolgar, 1979), stripped of the modalities of debate that characterized its development even while gaining impact and momentum as a broadly used vernacular category and organizing principle in small and large information technology endeavors alike.

A 'real world' for artificial intelligence

> Designing such a program requires commitments about what knowledge is and how it is obtained. Thus, some of the major traditional problems of philosophy arise in artificial intelligence. (McCarthy and Hayes, 1981 [1969])

Writing retrospectively in 1993 on the state of their field, Stanford computer scientists Edward Feigenbaum and Bruce Buchanan (1993) sought to characterize 'a shift in paradigm' in artificial intelligence research from 'one based on generality' to 'one that was knowledge-based' (p. 236). A program called the General Problem Solver (GPS) served as their exemplar of generality, while another program called Mycin served as their exemplar of the knowledge-based paradigm. In this section we track this transition via these two artificial intelligence projects, though we ultimately conclude that they represent poles for the logic of domains rather than any paradigmatic transformation. The former treated all domains as decomposable in the same way, while the latter served to consider whether domains may have a logic or style of reasoning of their own, even while both proffered an approach that was independent of any domain.

Domains are the 'real world' to artificial intelligence researchers and expert systems designers. The phrase 'real world' is liberally scattered throughout the AI writings of the 1960s and 1970s (and still today), often appearing in quotes, though sometimes without. As the developers of the expert system Mycin programmatically argued in their 1977 paper:

> Two recent trends in artificial intelligence research have been applications of AI to 'real-world'
> problems, and the incorporation in programs of large amounts of task-specific knowledge. The
> former is motivated in part by the belief that artificial problems may prove in the long run to be
> more a diversion than a base to build on, and in part by the belief that the field has developed
> sufficiently to provide techniques capable of tackling real problems. (Davis et al., 1977: 17)

If AI researchers were ready to tackle 'real world' problems in 1977, what had they been
up to until then? The critique was already well-established within AI circles that success-
ful applications of AI had focused on 'block worlds' or 'artificial problems', primarily
settings that operated with explicitly formulated rules readily amenable to the computa-
tional formalisms of the time, such as games (chess, checkers) and some highly mathe-
matized subfields such as physics that relied heavily on logical inference and deductive
reasoning. One concern was that the techniques of AI would only ever be applicable to
such artificial problems. But in the quote above, Davis et al. went further, suggesting that
turning to the 'real world' would challenge the field to encounter and manage that world's
full complexity.

A pervasive goal of AI was problem-solving, so too one of the most common modi-
fiers for domains at the time is 'problem', as in 'problem domains.' In addition to having
problems, domains were sometimes said to have the expertise, know-how or knowledge
to solve them.[2] An influential, but also controversial, formulation of problem domains
was articulated in Herbert Simon's (1973) well-known paper 'The structure of ill struc-
tured problems'. Simon (1973) identifies much the same challenge for AI research as
Davis et al. above, that is, transitioning AI from a focus on problems of logical and
deductive reasoning such as chess or theoretical physics to 'the real world of large prob-
lems' (p. 186). However, in contrast to the assertion that past work on artificial problems
might prove a dead end for AI, Simon (1973) strongly argued the exact opposite:

> There is no reason to suppose that new and hitherto unknown concepts or techniques are needed
> to enable artificial intelligence systems to operate successfully in domains that have these
> characteristics [ill structured problems]. (p. 181)

Simon asserted that any problem, whether well- or ill-structured, could be decomposed
using the same repertoire of analytical concepts and techniques, or 'formalism.' This line
of argumentation recapitulated much of Simon's work in collaboration with Allen Newell
and John Shaw, such as the development of the renowned AI program the General
Problem Solver (GPS):

> the core of GPS consists of some general, but fairly powerful, problem-solving heuristics. To
> apply these heuristics to a particular problem domain, GPS must be augmented by the
> definitions and rules of mathematics or logic that describe that domain … The justification for
> calling GPS 'general' lies in this factorization of problem solving heuristics from subject
> matter, and its ability to use the same heuristics to deal with different subjects. (Newell et al.,
> 2000 [1959]: 72)

As with Simon, 'problems' in GPS are specific to domains, whereas problem-solving
heuristics are general. AI researchers took inspiration from their model of the human
mind (Nilsson, 2009): In particular, minds are capable of solving many kinds of

problems, and so, any program that deserved the title 'intelligent' would also have to be general. GPS came to be successfully used for applications such as certain chess and logic problems, and solving the Tower of Hanoi. However, only a few years later GPS came to stand in for an extant provinciality in AI research that merely focused on well-structured domains. Would approaches such those of the GPS be limited to artificial problems, rather than the real world of ill-structured problems?

The language of the 'real world' was accompanied by an empiricist ethos: Knowing and representing the problems of the domains would involve some form of research rather than a logical or deductive arrival at their problems. John McCarthy (often attributed with naming artificial intelligence (Boden, 2008; Nilsson, 2009)) and Patrick Hayes put it this way:

> The right way to think about the general problems of metaphysics and epistemology is not to attempt to clear one's own mind of all knowledge and start with 'Cogito ergo sum' and build up from there. Instead, we propose to use all of our knowledge to construct a computer program that knows. (McCarthy and Hayes, 1981 [1969]: 6)[3]

The task at hand, then, was to scour the world for 'all of our knowledge' and represent it within computable formalisms. Or, cast programmatically: 'How are observations to be used to get knowledge about the world, and how are the other kinds of knowledge to be obtained?' and 'In what kind of internal notation is the system's knowledge to be expressed?' (McCarthy and Hayes, 1981 [1969]). These two questions – how to acquire knowledge from domains, and how to represent that knowledge in a computable formalism – become central challenge questions for the logic of domains thereafter.

While empiricism (or, more precisely, the assembly of extant knowledge) animated the discourse of expert systems in the 1960s and 70s, the methods of inquiry to capture that knowledge were rather haphazard. In the 1980s, discussions of methodology would flourish with the first formally articulated methods and support tools for 'domain analysis', and eventually the founding of focused journals such as *Knowledge Acquisition* in 1989 – discussed in the next section. But in the early 1970s, discussions of empirical methods for knowing the domains, or acquiring their knowledge, were usually interleaved into the discussion of the development of particular programs rather than a discrete topic to be addressed on its own.

Always accompanying commitments about the content of a domain was the question of how to 'acquire' or 'elicit' that knowledge, along with how to 'represent' or 'capture' it. Representing knowledge in a computable form requires forms of media for doing so; and so, in these years of AI we can observe the creation of a series of competing formalisms for computable knowledge representation (Poirier, 2018). In this milieu, formulations of what matters in a domain (almost invariably, problems and the knowledge to solve them), coevolved with techniques for acquisition and media for representation.

Here we return to Mycin. In his 1980 report to the US Office of Naval Research, Edward Feigenbaum (1980: 1) characterized expert systems as 'the applied side of artificial intelligence'. Feigenbaum himself had been a key developer of the expert system DENDRAL and the machine learning system Meta-DENDRAL, programs that analyzed mass spectral data to infer a likely chemical structure. Mycin borrowed many architectural features from DENDRAL (discussed below), but at a more conceptual level

Feigenbaum and Buchanan (1993) characterized DENDRAL as having 'set in motion a shift in paradigm in AI from one based on generality to one that was knowledge-based' (p. 236). As we have noted, they took GPS to be characteristic of the generality paradigm, while Mycin served as their exemplar of an expert system set in the knowledge based paradigm.

Mycin was initially developed as Robert Shortliffe's medical dissertation, in collaboration with artificial intelligence researchers Randal Davis, Bruce Buchanan and others in the computer science department at Stanford University (Berg, 1997). This expert system and its many descendants were to remain the research object of this lab of artificial intelligence researchers for over a decade. Mycin was intended to identify bacterial infections such as meningitis or bacteremia by having doctors input symptoms to the computer program. With sufficient and consistent inputs, Mycin would diagnose and then recommend an antibiotic treatment. The expert system itself was named after a common suffix for antibiotics: -mycin. Here we will briefly focus on the capture of domain knowledge in production rules, and then turn to the crafting of Mycin's domain independent offspring, eMycin.

Mycin is known as a rule-based expert system; it represented the knowledge of infectious disease specialists in if/then 'production rules', each with a premise and an action. For example,

> If (1) the infection is primary-bacteremia, and
> (2) the site of the culture is one of the sterilesites, and
> (3) the suspected portal of entry of the organism is the gastrointestinal tract,
> then there is suggestive evidence (.7) that the identity of the organism is bacteroides. (Davis et al., 1977: 21)

Each rule is a 'single, modular chunk of knowledge' (p. 21), in that it can be added, altered or removed, without requiring a rewriting of the entire knowledge base. This design feature was intended 'to accommodate a large and changing body of technical knowledge' (p. 17), or more generally a recognition that the domain is not static. Modularity is a commonly posited solution to the problem of changing domains. In addition to changing knowledge, the domain of medicine demonstrated features that distinguished it from the 'toy problems' of earlier AI applications:

> Since we want to deal with real-world domains in which reasoning is often judgmental and inexact, we require some mechanism for being able to say that 'A suggests B', or 'C and D tend to rule out E'. (p. 22)

The developers of Mycin characterized medicine as a domain of judgmental reasoning in which evidence can be incomplete, uncertain, or contradictory, and yet decisions must be made in a time-bounded manner. And so, rules also had to reflect the uncertainty of the domain, and Mycin's heuristics the real world procedures for decision-making in the face of uncertainty. For Mycin, the solution was to give each rule a certainty factor, or the 'numbers used to indicate the strength of a rule' (p. 22), and to annotate the system's output decisions a with an aggregated certainty factor (e.g. '.7' in the quote above). More generally, Davis et al. called for more study of 'models of inexact reasoning' (p. 29), which in addition to certainty factors included, for instance, the further

development of fuzzy logic. With the recognition that real world domains operate in the face of uncertainty, or using judgmental reasoning rather than predicate logics, one response was to develop formalisms that could reproduce or mirror those forms of reasoning. In the 'knowledge based approach' of Mycin, one consideration was that domains might have styles of reasoning, or logics, of their own.

The formal production rules were 'encoded' representations of domain knowledge, 'acquired' from experts themselves. In publications, very little is reported about the experts who contributed their time and knowledge to this enterprise, who were primarily doctors in the Stanford medical school. Even less is reported about the practical process of knowledge acquisition, with only very high-level descriptions of the prompts and tasks given to experts, such as having experts articulate premises and actions. As we noted above, this is reflective of a thin methodological repertoire for relaying the knowledge acquisition process, a repertoire that would later expand enormously; this is discussed in the next section.

However, Davis et al. (1977) reflect extensively about the limitations of their acquisition process. For instance:

> a fundamental assumption is that the expert teaching the system can be 'debriefed', thus transferring his knowledge to the program. That is, presented with any conclusion he makes during a consultation, the expert must be able to state a rule indicating all relevant premises for that conclusion. The rule must, in and of itself, represent a valid chunk of clinical knowledge. (p. 40)

These are indeed rather vast assumptions, and, as we noted above, they become the topic of an expanding literature and methodological repertoire focused on knowledge acquisition and representation, as well as the grist for critiques of the entire expert system research program (c.f., Collins, 1990; Forsythe and Buchanan, 1989; Schank and Jona, 1994).

We have thus far portrayed as distinct the approaches for Mycin and the General Problem Solver, but it their continuities that reveal a common style of organizing across the logic of domains. While the developers of these programs approached the domains differently, they shared the conviction that there is something beyond the domains that is 'general' or 'independent.' As we have seen, initially Mycin was domain specific, but a key feature was that, from the start, it was developed with the goal of supporting domain independence. This becomes explicit with eMycin or 'empty Mycin'.

What was eMycin empty of? Domain knowledge. eMycin is often described as the first 'shell' for expert systems. By emptying the program of its domain specific rules, what was left was a tool based on the 'domain independent core of the Mycin program' (Van Melle, 1980: 1). As with GPS, what was general about eMycin was its reasoning heuristics. This core or shell was then 'applied' to other domains, sustaining its reasoning heuristics, but thereafter filled with production rules specific to another domain.

How does one empty a program of its domain knowledge? According to the developers, Mycin had been designed to support such an activity from its inception by keeping separate the procedural knowledge base (domain) from the reasoning heuristics (independent). This architectural design consideration had been inspired by the General Problem Solver and adapted from DENDRAL: 'No other AI program, including DENDRAL, had been built using so much domain specific knowledge so clearly separated from the inference procedures' (Buchanan and Shortliffe, 1984: 11). Here,

the conceptual distinction between the domain specific and independent features was materialized in an architecture that separated knowledge from reasoning. However, as one developer of eMycin noted:

> Despite the stated design goal, there was much code in Mycin that referred to specific aspects of the infectious disease domain. These ranged from the simple cases of strings and other quoted text referring to 'the patient' (these were changed to refer to variable strings, or more generally to 'the case'), to instances of actual domain knowledge in the code. … In some cases this knowledge was reformulated into rules, while other instances became domain specific 'hooks' (Van Melle, 1980: 8)

This account hints at the kind of practical boundary work required to enact the distinction between domain and independent features. In this case, boundaries were drawn by, first, designing in the software a division between general heuristics and domain specific rules, and thereafter creating a shell by carefully rewriting user prompts and code vocabularies in a general language.[4] Following this, using eMycin as a shell, various AI researchers went on to create domain specific production rules in the interpretation of, for instance, pulmonary function measurements ('PUFF'); diagnoses of a range of psychiatric disorders and recommended drug treatments ('HEADMED'); and, with SACON, 'in a stronger test of domain independence, eMycin was applied to the non-medical domain of structural [engineering] analysis.' (Van Melle, 1980: 6). Even while divesting and repopulating all this domain knowledge, eMycin retained its etymological roots via its antibiotic inspired naming.

eMycin came to serve as an exemplary second pathway to domain independence. While GPS had been developed to be general, thereafter to be 'augmented' with domain specific content, Mycin began as a tailored domain specific application organized so as to facilitate division of domain from independent features; from this a domain independent shell was extracted.[5] In almost all cases in the 1960s and 70s (but certainly not in all figurations in this article), reasoning heuristics are the central features of what constitutes generality or domain independence, while problems and the knowledge to solve them constitute domains. The development of a general tool thereafter tailored to domains (e.g. GPS), or a domain specific tool thereafter stripped of its specificity (e.g. eMycin), became two common approaches in AI and beyond. But these are only two of the proffered ways for achieving domain independence; we explore others, such as 'domain analysis' in the next section, in which a feature is identified as common across domains and thereafter placed in circulation for general reuse across them.

With both GPS and eMycin, and more broadly in AI at the time, universality had been defined and architecturally materialized as an absence of specificity. This formulation thereafter propels a common dynamic in the logic of domains: Efforts to engineer generality or domain independence demand a concurrent empirical project that seeks out and fills in the missing quantity of specificity, i.e. the domains, or the 'real world'.

The next section tracks the logic of domains beyond artificial intelligence, with a focus on the mutating and expanding content of domains, modes of inquiry, and materializations in formal methodologies and tools. However, while in this article we leave artificial intelligence behind, AI has not left behind the logic of domains. We can find papers, tools and programs throughout the late 20th century, and right up the present, that

rely on this formulation, advancing it, challenging it, and offering new modes of inquiry and formalisms for representation.

Domain analysis: Formalizing approaches to the domains

In the first approach explicitly called 'domain analysis', a domain came to refer to the organizational setting of software use. While artificial intelligence researchers had set themselves a grand task — developing knowing, reasoning and problem-solving machines — software engineering researchers in the 1980s took on the arguably more mundane goal of facilitating the reuse of software. Perhaps mundane, but certainly not easy, software reuse remains a lively topic of research throughout the late 20th and early 21st centuries, and descendants of domain analysis initially developed by James Neighbors, Rubén Prieto-Díaz, Guillermo Francisco Arango and Peter Freeman at the University of California Irvine in the 1980s remains one key method promising successful reuse.

This section focuses on domain analysis to illustrate the drive to formalize methods of inquiry on and encoding of domains. The discussions that surrounded domain analysis are exemplary of goals to encode the logic in explicit method, toolkits and software suites. Such approaches seek to regularize, systematize or render easier the doing of domain analysis. Domain analysis is an example of the development of methodic and tool-based approaches to the logic of domains. By delegating inquiry and representation to prepackaged methodic approaches, knowing domains and generating pathways between them becomes easier to do, and perhaps also harder to think about; not quite black-boxed, but on a pathway to it.

Set within a broader set of concerns that historian Ensmenger (2012) calls 'software's chronic crisis', domain analysis sought to reuse general 'software parts' within and across specific domains. Drawing on the history of gun manufacturing, in his 1980 computer science dissertation James Neighbors argued for a 'parts and assemblies' approach to software, in which problems or software features were identified in order to produce general software 'parts' and standard approaches to their reuse, or 'assembly'. He called the approach Draco, and the process of identifying those common problems and tasks domain analysis: 'The concept of domain analysis is introduced to describe the activity of identifying the objects and operations of a class of similar systems in a particular problem domain' (Neighbors, 1980: 1).

As with domains in knowledge-based expert systems, domains in software engineering are empirical objects that must be investigated to be known, and thereafter represented. Neighbors identified his research as sitting at the intersection of AI and software engineering, and drew heavily from the expert systems literature and AI more generally. But while Neighbors made the genealogical lineage clear, he significantly transformed the concept of a domain, at times making it sound like the object of social scientific fieldwork, as in his example of developing software tools to aid travel agents:

> [Domain analysts] would go out to travel agent offices and study the activities of travel agents. A model of the general activity of being a travel agent would be formed and the objects and operations of the activities identified. At this point, the analyst of the domain of travel agent systems would decide which general activities of a travel agent are appropriate to be included in travel agent systems. (p. 6)

A notable difference from what we have seen before lies in what domains are 'made of.' In domain analysis, the object of interest is less 'knowledge' and more the 'activities' and 'objects' in a domain. A second difference is a more capacious understanding of expertise. AI researchers had focused largely (though not exclusively) on scientific contexts, but with domain analysis in software engineering, expertise is a property of business (e.g. a travel agent), or really any group using software.

But what most sets domain analysis apart from what we have seen before is the focus on developing a routine, repeatable method for inquiring upon and representing the domains. Neighbors posits several specialists who were to embark on the task, who he called domain analysts, modelers and designers. Figure 1 is drawn from a paper published in 1989, in which the process of domain analysis is cast generally. On the left are 'users of similar systems', in the middle various 'domain modelers' acting as translators, and on the right the Draco repository itself that collects the outcomes of iterations of domain analyses. The domain modeler's goal is to identify commonalities across similar systems, encode or extract these as general software features, and place those features in a repository for reuse.

A careful inspection of Figure 1 reveals that users of similar systems (top-left) are portrayed as happy smiling characters, while the domain analysts (middle) appear ambivalent. This is no accident. Neighbors emphasized the investment and difficulty of
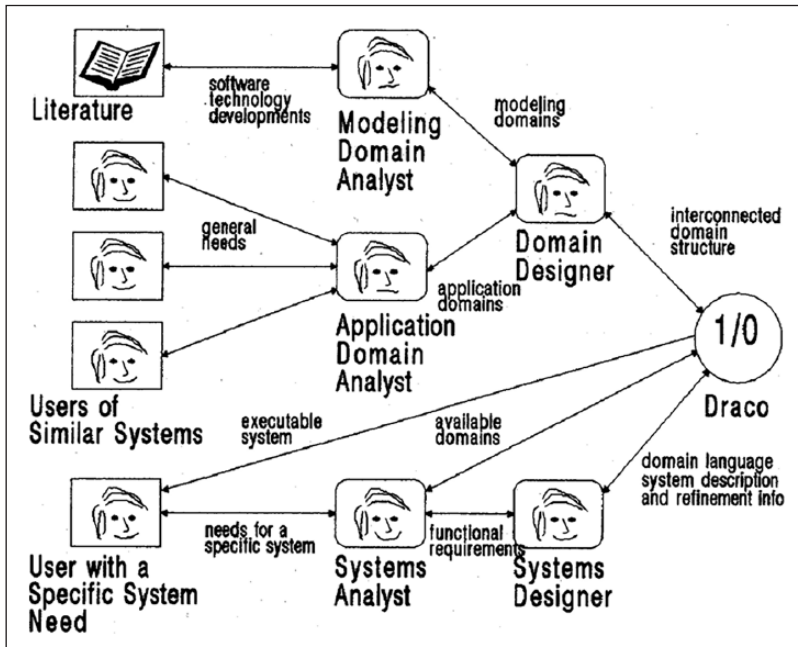
Figure 1. The process of analyzing domains (Neighbors, 1989), moving from specific (left), to domain modeling (middle), the general Draco repository (right), and specific reuse (bottom). Domain analysis is hard (thus the frowny faces for analysts) but facilitates reuse (thus smiling faces below).

conducting domain analysis, and his methodology generally reflects a mounting drive to formalize inquiry and capture:

> Domain analysis describes a range of systems and is very expensive to perform. … Domain analysis requires a *craftsman* with experience in the problem domain. (p. 7, emphasis added)

The craftsman is an intriguing figure for the logic of domains. As Neighbors notes, 'a craftsman gains enough experience to create a problem domain' (p. 8). Inscribing the logic of domains in a single individual, the craftsman may themselves *be* a domain expert but with the skills to identify and encode the domain in a computable formalism.

   In the broader trajectory of the logic of domains, boundary-spanning figures such as the craftsman are a commonly proffered solution to the problem of transcending the specificity of domains. By various names, such figures populate each of the manifestations recounted in this article. Arguably, Shortliffe himself was that figure in the development of Mycin, gaining a dual expertise in medicine and in formulating infectious disease as computable rules. As eMycin's developers came to work with more and more domains, they articulated a role for an 'intermediary' or 'knowledge engineer':

> In our experience, the model for this process has been that a person ('knowledge engineer') familiar with eMycin's representation interacts with the expert, elucidating the task to be performed and the particular knowledge needed for reasoning in the domain. (Van Melle, 1980: 149)

Representing the domain in computable rules requires understandings of the distinct technicalities of both the domain and the formalism. Later in this article we discuss 'the bilingual', 'the pi-shaped person', and efforts in Cyberinfrastructure to systematically develop a workforce of such boundary-spanning figures. Generally speaking, the hope is that such boundary-spanning figures will be able to sustain fidelity to esoteric expertise while transcending its specificity, faithfully encoding the domain in computational formalisms.

   It is worth taking a moment to contrast the approaches to developing generality or independence across expert systems and domain analysis. With eMycin, the general shell for expert systems, that independent software artifact was moved from one domain to the next, and was 'filled' with that domain's specific knowledge at each stop. On its own, it is 'empty', in that domain content was evacuated from Mycin to create a domain independent shell, thereafter of potential use in any domain. In contrast, in domain analysis a feature is identified as common at a specific site, that feature is characterized in a formal domain representation (or software 'parts and assembly') and then deposited in the repository for reuse. Much like eMycin, the Draco repository is initially 'empty' of domain content. Over time, and across many domain analyses, the repository collects more software 'parts and assemblies' as further resources for software reuse. In that it is applicable to all domains, eMycin is general. In the case of domain analysis, specific software is rendered general and able to circulate across domains by placing that software in a repository. While figured differently, both approaches exhibit the telltale

features of the logic of domains: domain specificity, independence or generality, and the goal to circulate resources across domains.

Multiple investigators have further sought 'to formalize the domain analysis process, and to develop technologies and tools to support it' (Prieto-Díaz, 1990: 48). For instance, Feature Oriented Domain Analysis (FODA), developed by researchers at Carnegie Mellon University, extensively describes multiple stages of customer engagement, research, identification of general features, modeling and representation, and formalization. Much like the goals for software in domain analysis, such methodologies seek to render domain analysis itself into a portable 'off the shelf' approach for use in any context where software reuse may be desirable. A 2009 review article identified 31 software tools developed to support domain analysis. Such tools encode methods and categories in order to facilitate the process of identifying and circulating general software components, thus supporting 'a repeatable process, and … reuse of higher level life cycle artifacts' (Lisboa et al., 2010: 2).

Efforts to formalize methodology also speak to the challenge of actually analyzing domains, for a recurrent theme in domain analysis has been the desire to transform the approach 'from art form to engineering discipline' (Arango, 1989: 152). The many extant methodologies and tools speak to the success of formalizing the broader arc of domain analysis, even while other aspects have proven systematically challenging. In virtually every effort to systematize domain analysis, the authors have found themselves reminding readers of key challenges: 'knowledge about a problem domain is often implicit and nonformal, while reusable information must be usually represented explicitly and formally' (Arango, 1989: 154). Prieto-Díaz (1990), in his effort to codify domain analysis, remarked on the seemingly bottomless specificity of domains and consequent need to tailor analytic methodologies, once again calling for a boundary-spanning figure: 'A domain analyst is expected to be a person of all trades' (p. 52). Similarly, even while calling for domain analysis to become an engineering discipline, Arango (1989) lamented: 'Potentially, there is a need for a large number of practical domain analysis methods, at least as many as there are types of reuse' (p. 159). In short, even as these researchers sought to codify the overarching process of domain analysis, they noted how the heart and core of analyzing and representing a domain might prove recalcitrant to generalization, or, itself domain specific.

Domain analysis in software engineering remains an academic research enterprise.[6] As such, even while seeking to package up their approach in methodic or software based approaches they continued, and even promoted, a speculative discourse on what domains 'are', 'have' or are 'made of', and what may be the nature of intermediation across domains. Methodic, formal or tool-based approaches to the logic of domains thus serve to facilitate the analysis and representation of domains, along with their intermediation, but without fully sealing into a black box such questions as: *What are domains? How do we know them? In what formalism do we capture them?* In the next section we explore how this discourse has been picked up in science policy, technological development projects and vernacular talk, and therein has largely lost this speculative quality, even while serving as a *de facto* organizing principle.

From technical to vernacular: The logic of domains as an organizing principle in science policy

By necessity, our tone shifts markedly in this section as we focus on how the logic of domains has become a vernacular organizing principle. The general trajectory we observe is a shift from a series of academic research programs (i.e. expert systems or domain analysis) to an organizing principle that takes the logic as its implicit starting point. McCarthy and Hayes' call to examine epistemological commitments is far more challenging in vernacular manifestations of the logic of domains, for in science policy and in everyday talk commitments are rarely made explicit and the concept of domains is treated casually, as a matter of fact, rather than speculatively. This is likely intentional as, for instance, calls for funding proposals are broadly worded to encourage a variety of submissions. But without explicitly stated commitments, so too fall out many of the debates we characterize for the logic of domains in this article. It is in this sense that we use the term 'vernacular', and it is in this sense that the logic of domains has become an organizing principle: Domains and the need for their intermediation act as received starting points to arrange some technological endeavor of scientific and engineering research, to organize a heterogeneous collaboration, or to build tools and infrastructure.
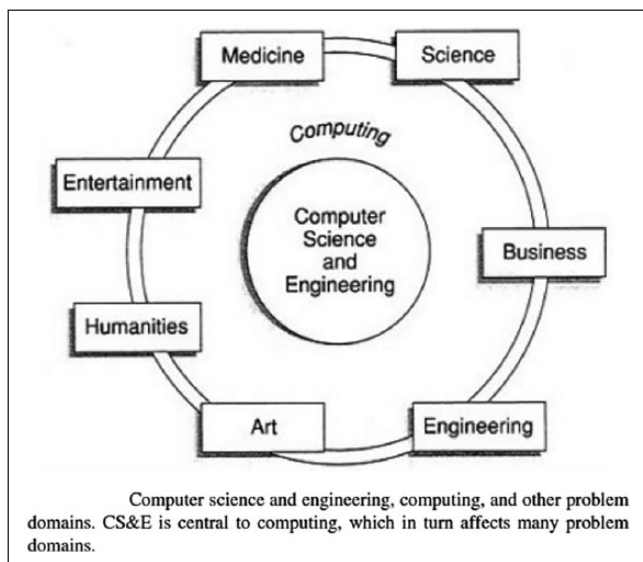
Our inspections of policy are exclusively focused on US science policy and organizational forms — while there is strong evidence that some form of the logic reaches well beyond, we limit ourselves to the settings we understand best. Drawing on the language of the *Computing the Future* report, here the logic of domains becomes an 'engine for progress', as funding opportunities are organized as collaborations of computer (information or data) scientists in tandem with domain experts, promising advance and utility for both.

The late 1980s is often described as an 'AI winter', a period in which the ambitious claims of AI, and of computer science more broadly, came under heightened scrutiny in the academy and greater suspicion within business (e.g. Boden, 2008; Nilsson, 2009). The result was a decline in public and private funding, and esteem. One response was to develop a new strategy: Rather than buttress the disciplinary boundaries of computer science, that discipline sought to embrace collaborations with other scientific and engineering practitioners, and with business and government, i.e. the domains – see Figure 2. In the words of the *Computing the Future* report, computer science and engineering (CS&E) *should*:

> recognize that intellectually substantive and challenging CS&E problems can and do arise in the context of problem domains outside CS&E per se. CS&E research can be framed within the discipline's own intellectual traditions but also in a manner that is directly applicable to other problem domains. (NRC, 1992: p. 4)

Broadly read and influential, *Computing the Future* articulates the logic of domains as one *de facto* operating procedure for small and large applied computing ventures, promising insights for domains and basic advances for computing: 'CS&E can thus be an *engine of progress* and conceptual change in other problem domains, even as these domains contribute to the identification of new areas of inquiry within CS&E' (NRC,

Computer science and engineering, computing, and other problem domains. CS&E is central to computing, which in turn affects many problem domains.

Figure 2. Drawn from the *Computing the Future* report, caption in original (NRC, 1992).

1992: 4, emphasis added). We call this the *all boats will rise thesis* because it is often explicitly stated using the 'a rising tide lifts all boats' proverb in ventures that call for the collaboration of computation (information or data) science and the domains. Perhaps implicit throughout the figures we have examined, beginning in the 1990s, the all boats will rise thesis becomes a mainstay in science policy circles associated with the logic of domains.

Beyond this new strategy, *Computing the Future* articulates two common tensions that have haunted computational practitioners adopting the logic of domains: the threat of lessened academic prestige by being 'applied researchers', and the danger of being treated as a 'service science'. As with most academic scientific enterprises, the label 'applied' in the computing sciences has traditionally been associated with practical and site-specific outcomes, while 'basic' research carries with it promises of generality. More broadly, since the logic of domains has always been concerned with engaging some facet of the real world, the logic has been associated with applied research from its beginnings; for example, recall Feigenbaum's characterization that expert systems are 'the applied side of AI'.

The C*omputing the Future* report vociferously rejected the basic and applied distinction for the computing sciences, instead asserting that working with the domains is a pathway to basic insights, generality and perhaps universality:

> at least within CS&E, the traditional separation of basic research, applied research, and development is dubious. Given the way research in CS&E is practiced, distinctions between basic and applied research are especially artificial, since both call for the exercise of the same scientific and engineering judgment, creativity, skill, and talent. (NRC, 1992: 5)

The distinction between basic and applied has never been a firm one, and Gieryn (1983) and Calvert (2006) have shown that it is best thought to be a strategically deployed boundary. But, as Gibbons et al. (1994) have also shown, the late 1980s and early 1990s witnessed a widescale revisioning of the importance of applied science, and of collaborations across academic, industry and government sectors.

Being applied, however, bears more dangers than whether it can achieve generality. Applied computing also carries the risk of sidelining novel research, casting computer scientists as 'mere' technicians, software or web developers, or some sort of infrastructural plumber. The danger is quite real, as there is marked tension between the labor of research that pushes the limits of computation and the enormous task of developing stable, usable and sustainable systems (Ribes and Finholt, 2009). *Computing the Future* suggested a structural solution — one that has itself become a *de facto* organizing principle — ensuring a protected position for novel computational research even while continuing to promise utility for the domains:

> It is essential that such work be done by investigators from CS&E and other disciplines and areas who regard each other as intellectual equals … maintain[ing] both an understanding of the future state of the art in computing and an appreciation of the *real problems* in the application domains. One way of ensuring true collaborative work is to consider only proposals where principal investigators are drawn both from CS&E and some other discipline or area. (NRC, 1992: 146 emphasis added)

By recommending that leadership be composed of computing *and* domain scientists, this funding structure sought to ensure that the research interests of computing were not subjugated to goals for providing useful domain resources.

Following the *Computing the Future* report, this funding model became an organizational mainstay for computing funding, particularly through its 'applied' offshoots such as the National Science Foundation (NSF) Digital Libraries program and the Information Technology Research (ITR) funding line, Cyberinfrastructure and, most recently, Big Data, and data science more generally. Here we will focus on Cyberinfrastructure and data science funding ventures at the NSF.

Cyberinfrastructure presented a unique 'infrastructural' formulation of the logic of domains. Figure 3 is drawn from the influential *Revolutionizing Science and Engineering Through Cyberinfrastructure* NSF report, more commonly called the *Atkins Report* following its chair (Atkins et al., 2003). This image of organization presents a vision for infrastructure across the sciences and engineering, tailored for the domains but designed for generality across them. Drawing on a common visual trope of a 'stack', elements higher on the diagram are closer to the user (i.e. domain scientists) while those elements lower on the stack are more general, universal or fundamental. Thus, domain specific applications are undergirded by generic applications and even more generic core information technologies.

Drawing on rhetoric similar to that of the software crisis in domain analysis, the Atkins Report asserted that Cyberinfrastructure should avoid the waste of recreation, and embrace modularity and reusability:

> Commonalities across science and engineering disciplines must be captured. Absent appropriate levels of coordination and sharing of facilities and expertise, there would be considerable duplication of effort, inefficiency, and excess costs. (Atkins et al., 2003)

Commonalities across domains are the bailiwick of core information technologies even as Cyberinfrastructure would continue to require the development of domain specific applications. Recapitulating the concerns of *Computing the Future*, the *Atkins Report* poses the development of infrastructure as a balance between domain and computational advances:

> If the organization is weighted too heavily toward the domain scientists, the focus overemphasizes procurement of existing technologies, and computer scientists become viewed as 'merely' consultants and implementers. If the weight shifts too heavily toward computer science, the needs of end users may not be sufficiently addressed, or effort shifts too heavily toward creating new technologies with insufficient attention to stability and user support. (Atkins et al., 2003: 50)

Such programmatic policy formulations of the *all boats will rise thesis* cast a dual responsibility for technology development efforts: i.e. serving to push forward research on novel computational capacity while simultaneously capacitating the domains in computational technique.

This dual responsibility is recapitulated throughout the vision of Cyberinfrastructure, such as with the people who will build it, and in the architecture itself. For example, the report called for the education of a new kind of workforce, involving a systematic training of boundary-spanning figures who are simultaneously domain and computational experts: 'These individuals have expertise in a particular domain science area, as well as considerable expertise in computer science and mathematics' (Atkins et al., 2003: 26). In addition, the report also inscribes the logic of domains into the very technical architecture of Cyberinfrastructure: 'Applications are a hybrid case with shared responsibility between technological and disciplinary programs.' Applications thus have a dual job of meeting the needs of the domain (upwards in Figure 3), and a 'responsibility' to a generic architecture undergirding connections across the domains (downwards in Figure 3). Goals for spanning the boundaries between domains and what is beyond them (or with Cyberinfrastructure, 'below them') were thus recapitulated at each scale, or what Ribes (2018: 527–528) has called the fractal or holographic properties of domains: the individual as both a domain and computational expert; a dually responsible application layer; and Cyberinfrastructure as intermediating all the scientific and engineering domains.
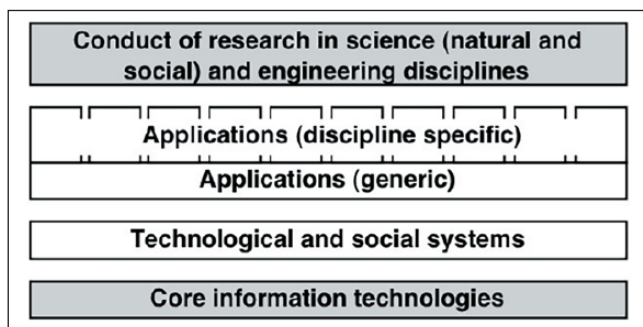
Figure 3. Cyberinfrastructure architecture diagram (Atkins et al., 2003). Specific disciplinary applications are undergirded by generic infrastructure: '[generic] applications are a hybrid case with shared responsibility between technological and disciplinary programs.'

The Cyberinfrastructure program primarily targeted what is perhaps the most commonly articulated feature of domains today: they have 'real world' data. In general, projects dubbed Cyberinfrastructure sought to facilitate the discovery, circulation and reuse of data across their targeted domains. This interest in data has magnified in the 2010's with Big Data and data science.

Turning to data science, a recent call for proposals at NSF sought to foster the development of a theoretical core for that nascent field: *Transdisciplinary Research in Principles of Data Science*, or TRIPODS. From its 2016 call for proposals, the TRIPODS funding program 'aims to bring together the statistics, mathematics, and theoretical computer science communities to develop the theoretical foundations of data science' (NSF, 2016). It is these three disciplines that are solicited to develop TRIPODS centers, marking them as the domain independent experts of data science.

Scrolling down to the Frequently Asked Questions (FAQ) released with the TRIPODS call for proposals, one finds the question 'What is the role of domain scientists? Can a domain be the primary focus?' The answer from the NSF (2016) is worth quoting in full here:

> Domain scientists should serve as a resource for a TRIPODS institute to maintain connectivity to *real-world* issues and ensure relevance. That said, while practical applications serve as drivers, foundational work should be broadly applicable. Bear in mind that proposals will be reviewed by a broad range of mathematicians, statisticians, and computer scientists, and thus focusing too narrowly on a particular domain may not have the broad appeal to be considered competitive in the context of the goals of the TRIPODS program. (emphasis added)

TRIPODS simultaneously defines the insides and outsides of a theoretical core for data science, and thus the recipients of this funding allocation as the domain independent fields of mathematics, computation and statistics.

However, no formulation of the logic of domains is complete without a role for the domains. Following the awarding of 12 TRIPODS centers in 2017, NSF rolled out its follow-on funding program *Partnerships between Science and Engineering Fields and the NSF TRIPODS Institutes*. More commonly known as *Tripods+X*, in this formulation participating groups are 'referred to as "Field X" or "domain science & engineering"' (NSF, 2018), with 'X' evoking its traditional role in equation-solving as *any* unknown variable. Tripods+X projects are thus cast as 'domain applications' of fundamental or general data science theory, with, in 2018, projects funded to investigate a wide header of topics such as chemistry, climate science or supply chain optimization.

Projects funded by TRIPODS and TRIPODS+X grants thus return to the research, method and tool development activities we have recounted in previous sections. Science policy that adopts a *de facto* formulation of the logic of domains itself serves to generate new avenues for novel research on domains and independence, along with technology development efforts that serve to facilitate the analysis, representation and intermediation of domains. Inscribed in science policy, such funding opportunities indeed do become 'an engine' for the logic of domains, propelling it anew. However, whether this results in conceptual progress for both computation and the domains has received very little empirical treatment, as discussed below.

While the figures we have presented in this article are roughly arranged in a chrono-logic manner, later figures have not superseded earlier ones. Quite the opposite: Science policy that organizes (and funds) projects by the logic of domains opens new avenues for research and tool building. Today, there are many lively threads of the logic that explore, systematize and delegate, novel technical formulations for how to know and represent the domains. Neither wholly formalized nor wholly vernacularized, all these threads remain in active discourse, operating in parallel, in tacit coordination, and sometimes in agonistic encounters.

Recurrent challenges

By way of discussion, here we offer a synthetic view of the common set of problems that have accompanied formulations of the logic of domains. No single actor articulates all these problems; instead, they appear sporadically across literatures while also repeating systematically. We parse these into four repeated articulations of problems: ontology, inquiry and representation, accountability, and the relationship of the general and the specific. We offer these statements of problems, first, as responses to the vernacularization of the logic, and second, as a means to set forth a program of research that will inspect the practice and consequences of the logic of domains.

The ontological problem: Are domains real? What are they? Are they the right way to approach the division of knowledge or expertise?

The use of 'domain' reveals several concurrent referents: world, expert and representation. The term is perhaps most commonly used to refer to the group of experts concerned with some worldly matter; such as, with Mycin, doctors diagnosing infectious illness. However, sometimes domain also refers to the representation of those experts; with Mycin, the knowledge base or collection of production rules. Finally, domain also refers to the world itself; with Mycin, infectious disease and its treatment. As computer scientist Phil Agre (2002) has noted:

> Even in domains that involve physical objects, it is common for AI people (and computer scientists in general) to employ the same words to name both the representations in a machine and the things that those representations represent. (p. 134)

On occasion, some investigator has set out on a Kantian endeavor to disentangle these, treating world, experts and representation discreetly. However, even in the most ardent such bounding efforts, usage tends to slip. More broadly, in talk and writing, the term 'domain' can refer to one or more of: a facet of the world, the experts concerned with that facet, and representations of that expert group. In some sense, then, all three of these are the 'real world', as measured against domain independence, which is not.

Some approaches take domains to be self-evident, ready-made or 'out there' while others treat domains as constructs: 'The domain is constructed in and through the process of planning, design and construction of a particular [system]' (Albrechtsen, 2015: 559). Similarly, some, perhaps most, have taken an emic approach to defining the contours of

domains, 'Each domain … is limited by a boundary that defines its scope. The borders define what objects, operations, and relationships belong to each domain' (Prieto-Díaz, 1990: 51). From such investigative approaches, domains come to mirror disciplinary formations, since when asked what counts as their domain, academic experts will often point to their discipline.

Others have taken an etic approach to the investigation of domains, such as with social network analyses of co-authorship or citation, '[s]ince there is no established, widely accepted "best grouping", we determined to derive our own from observed relationships … (i.e. their co-citation in articles)' (Porter and Rafols, 2009). This form of inquiry often reveals domains that do not mirror disciplinary identities. Such stances certainly have consequences, but the intellectual span of the logic of domains appears capable of operating across these seemingly disjunctured epistemological and ontological commitments.

The most persistent and evolving threads of inquiry have routinely asked what domains 'have', what is their 'content', or what are they 'made of': e.g. problems, knowledge, or logics. With the logic of domains such questions are never rhetorical, as they drive the adoption of methods of inquiry and media of representation for domains.

The problem of inquiry and representation: How should domains be known? How should they be represented? Are some domains more amenable to inquiry and representation?

Whatever theoretically significant features domains are taken to 'have' need be reflected in what modes of inquiry 'acquire' and what media of representation can 'capture'. As the potential content of domains has expanded, one response has been to adapt the methods of inquiry and representation from other fields (e.g. interviews, fieldwork, social network analyses) and to develop novel formalisms, such as for judgmental reasoning or domain specific logics.

A recurrent concern is whether the available approaches to inquiry and representation may forestall engaging some domains altogether. For example, the developers of Mycin recounted the limits of their method to represent knowledge as procedural rules: 'Not every domain will support this. It appears to require a field which has attained a certain level of formalization, which includes perhaps a generally recognized set of primitives and a minimal understanding of basic processes' (Davis et al., 1977: 30). If this is the case, what happens to those domains that have not 'attained a level of formalization'? Or, put less teleologically, are some domains less amenable to the available forms of inquiry and representation? Does domain analysis demonstrate a *preference* for certain styles of domains, and if so, will some domains systematically fall out of promised computational benefits? Some investigators have taken a stronger stance and suggested intervening on the domain to encourage formalization: 'A concern in domain analysis research is that we cannot wait indefinitely for a natural evolution of a domain but must stimulate its maturation process through a systematic process' (Prieto-Díaz, 1990: 49).

Debates about the nature of domains have thus been coupled with strategies to expand forms of inquiry and representation that mirror that nature, and, occasionally, efforts to

prod the domain to remake itself in the shape of whatever representational formalism or epistemic ideal is held at the time. A more subtle recurrent concern has been whether analysis and formal representation itself may transform a domain, and whether that is desirable or presents a fundamental challenge for the faithfulness of representation. However, as we elaborate below, essentially no research has systematically inspected the consequences of organizing by the logic of domains.

Problems of accountability: Are representations faithful? Who will arbitrate?

Formalisms often encode knowledge in ways that are inaccessible to the experts they seek to represent. Who are the arbiters of faithful representation? If the answer is experts themselves, then the next question is, 'How can we represent knowledge in a way that it is easily understood by humans and also machine processable?' (Prieto-Díaz, 1990: 52).

We have not followed the thread of the logic that has sought to automate the representation of domains – such as with machine learning – but such approaches have been present from our first figuration. Writing in 1977 about the advantages of Mycin's procedural rules over the machine learning system meta-DENDRAL, the authors extolled the virtues of symbolic AI:

> a system capable of handling an interactive dialog, and one which was not a 'black box'. This meant that it had to be capable of supplying coherent explanations of its results, rather than simply printing a collection of orders to the user. This was perhaps the major motivation for the selection of a symbolic reasoning paradigm, rather than one which, for example, relied totally on statistics. (Davis et al., 1977: 17)

Such long-standing discussions foreshadow contemporary concerns with the accountability of algorithms or the black-boxing of machine learning (Burrell, 2016; Kroll et al., 2016). Today, machine learning and other semi-automated approaches to representing the domains seems poised to explode (c.f. McCallum et al., 1999), and offer novel promises and dangers for representation, intermediation and accountability.

We have also noted recurrent boundary-spanning figures as the proffered solution to faithful representation and the transcending of domain specificity, such as the 'craftsman' of domain analysis or the 'knowledge engineer' of expert systems. A vivid figure in contemporary data science is the 'π-shaped person', with each leg of pi representing a distinct skill set and the squiggle between them the capacity to translate across these (Ribes, 2018). In 2018, the president of MIT University, L. Rafael Reif, proffered a new figure: the bilingual. In his announcement promising an investment of one billion dollars for a new college focused on artificial intelligence, he stated that the goal of the unit is not only to train a new generation of computer scientists, but also to 'educate the bilinguals of the future' (Lohr, 2018), with bilinguals defined as people in fields like biology, chemistry, politics, history and linguistics who are also skilled in the techniques of modern computing that can be applied to them.

Apprehensions about the interpretability of formalisms or representations, and recurrent boundary-spanning characters, both speak to the concern that representing domains

is epistemically challenging, and that evaluating representations is itself doubly technical, i.e. domains are often esoteric, and in a completely different way, representations of domains too are esoteric.

The relationship of the general and the specific

A consequential philosophical element of the logic of domains is the posited relationship between domain specificity and independence (or generality, universality and intermediation). It has often been suggested that the computing sciences are not empirical, and in that sense are more akin to mathematics than, say, biology or geology. But domains are precisely that, one of the long standing if only partially acknowledged epistemic objects of the computational or data sciences. As with any epistemic object, domains have displayed an emergent and recurrent trajectory, at times recalcitrant to characterization and manipulation, at other times leading to great practical advances. Epistemic objects are generative, or, as Rheinberger (1992) puts it, they are 'question machines', recurring anew at each investigation. Domains have proven to be question machines for decades, even as they have also served engineering goals by grounding computing applications in the 'real world'.

   We (and occasionally the actors) have cast the logic of domains as occurring in a rough spectrum that, on the one end, treats domains as fully decomposable using a common analytic approach or formalism, and, on the other end, treats domains as distinct, arguably with logics of their own, but still available to intermediation. Respectively we call these *representationalist* and *irreductionist* approaches (Latour, 1988; McCarthy, 2017; Monea, 2016). Herbert Simon has stood in for the representationalist end of the spectrum, with his (in)famous assertion that all problem domains, well- or ill-structured, can be approached in the same manner. This kind of assertion varies across figurations of the logic, but is recurrent. For example, in an effort to establish metrics for the representational capacity of computational ontologies, Wand and Weber (1993) proffer the concept of ontological expressiveness: formalisms 'are ontologically expressive if they are capable of describing all real-world phenomena completely and clearly' (p. 217).

   The irreductionist end of the spectrum is more nebulous, but is recurrently expressed as the possibility that domains may have logics of their own (such as 'judgmental reasoning') or the consideration that distinct methods and formalisms may be required for particular domains (e.g. as with domain analysis above). Star (1989) perhaps most clearly articulates the irreductionist position in an explicit critique of Simon's classic paper:

> Instead of a search for a logical Esperanto, … we should search for an analysis of objects. Problem-solving … produces workable solutions that are not, in Simon's terms, well-structured. Rather, they are ill-structured: they are inconsistent, ambiguous, and often illogical. Yet, they are functional and serve to solve many tough problems. (p. 51)

Rather than positing a singular base formalism or procedure across the domains (a 'logical Esperanto'), the *tertium quid*, which Star (1989) calls a boundary object, is: 'abstracted from all domains, and may be fairly vague. However, it is adaptable to a local site precisely because it is fairly vague; it serves as a means of communicating and cooperating

symbolically' (p. 49). This formulation sustains key features of the logic of domains: There are domains, they are specific and heterogenous and yet it is possible to translate or connect across these. But, as Star (1989) asserts, 'unlike Turing's universal computer, the creation of boundary objects both respects local contingencies and allows for cross-site translation' (p. 51).

Representationalist approaches appear to suggest that computational models can stand in for, even replace, domain expertise. Expertise appears as something to be extracted or automated, 'stretching whatever discourse they find upon the ontological grid that is provided by their particular design methodology' (Agre, 2004: 8). In such a model the expert is ultimately disposable, and the specificity of the domain is something to be overcome. Contemporary manifestations of this pole include arguments such as that big data analytics or machine learning will dispense with the need for (domain) theory altogether (c.f. Anderson, 2008; Slezak, 1989). In contrast, irreductionist approaches tend to take a realist stance to the heterogeneity of expertise, knowledge, or domains more broadly; in some form they tend to recognize that, say, infectious medicine may require epistemologies and methods distinct from those for, say, understanding geologists, or travel agents. Irreductionist figurations of the logic speak to a desire for approaches to heterogeneity (of knowledge, expertise, methods or data) that do not resort to overcoming difference by reducing it to a single underlying logic or decompositional heuristic.

While the nature of the relationship domain/independent is a theoretical one, we have sought to show how such theoretical commitments are materialized in the development of systems, representational formalisms, and domain inquiry methodologies or software packages. There are many additional posited relationships between generality and specificity than we have been able to discuss here, and we expect that further axes of difference will be necessary to characterize them.

Future research

From the perspective of these four recurrent sets of problems, the vernacularization of the logic of domains and its *de facto* status as an organizing principle is troubling. In these literatures, reports and calls for proposals, theoretical commitments remain opaque, and claims for promised benefits remain uninspected. For example, there is no scholarly literature that has systematically examined the *all will boats will rise* thesis: Does organizing by the logic of domains necessarily act as 'an engine for progress and conceptual advance' for both computing/data science and the participating domain? The assertion has proceeded as a token of faith, as a programmatic assumption, and as an organizing mythology. What scholarly literature does exist on the topic suggests systematic difficulties. For instance, Ribes and Finholt (2009) have noted the tension between developing novel computational capacities and offering stable infrastructure: the 'demos' or 'proofs of concepts' that serve as intellectual contributions for applied computing research are a long way from the usable, useful or sustainable tools or infrastructures that can reliably serve domain users. The gap is a formidable one, requiring care, labor and expertise that are not always (or even rarely?) considered in projects organized by the logic of domains. It seems clear that *all boats will rise* should not be considered a necessary or automatic

outcome of organizing under the logic of domains. Stated more strongly, without careful social organization and investments in usability and the sustainability of systems, the most likely outcome is that all boats will not rise.

A key topic for future research, then, is to examine the practical and institutional organization of endeavors organized by the logic of domains. The disconnection from intellectual reflection that accompanies pithy phrases like 'collaborations of domain and data science' renders theoretical or epistemological commitments opaque. As a result, alongside pragmatic investigations of the organization of the logic of domains, reflexive and empirical-philosophical investigations are needed; these include an evaluation of the distribution of resources and benefits across domain and independent fields, and an inspection of the consequences of formalization and intermediation for activities cast as 'domains.' Such questions return us full circle to our starting point with McCarthy and Hayes' 1969 call to inspect 'commitments about what knowledge is and how it is obtained', a question that was once central to AI, subsequently marginalized, but remains of inarguable consequence for any endeavor organized by the logic of domains.

Funding

Notes

1. The first figuration in this paper is not the 'beginning' of the logic of domains. The logic is observable before then, though less recognizably. For obvious reasons, before the 1960s the logic figures less as about developing computational systems, and more about logic, analytic philosophy and systems theory. All genealogical investigations demonstrate this quality: beginnings with receding horizons.
2. What AI researchers took to be the nature of knowledge at this time involves its own set of intricate debates, and attendant intellectual camps that formed around, for instance, whether expertise should best be thought of, and represented as, 'declarative' (i.e. what people say they know) or 'procedural' (i.e. know-how, or if/then rules). See (Adam, 2006; Winograd, 1975).
3. Reflecting on his first encounters with artificial intelligence researchers in the 1970s, philosopher Hubert Dreyfus sought to more precisely characterize their philosophical genealogies:

   They had taken over Hobbes' claim that reasoning was calculating, Descartes' mental representations, Leibniz's idea of a 'universal characteristic' – a set of primitives in which all knowledge could be expressed – Kant's claim that concepts were rules, Frege's formalization of such rules. (Dreyfus, 2007: 1137)

   To these we would add the influence of the mathematical theory of information, the universal Turing machine, and the algorithm as a central object for computer science.
4. See Ribes and Bowker (2009) and Suchman and Trigg (1993) for practical accounts of what such generalizing work might look like; and see Forsythe and Buchanan (1989) for a critical account that emerged from the collaboration of an ethnographer and one of the developers of Mycin.
5. The approach of developing a domain independent shell was successful in that it inspired many similar efforts, but it was also the object of critique. For example, Schank and Jona

(1994) called out the 'misconception that plagues expert system shells, what we call the "I'll build the architecture so all you have to do is put in the knowledge fallacy."' They instead argued for the primacy of mapping domain specific knowledge over general architectures. Within knowledge representation circles, this debate has been rehearsed again and again.

6.   On several occasions there have been efforts to commercialize expert systems and domain analysis such as via academic consulting services or through creating new companies. We do not trace such efforts here.

ORCID iD

David Ribes https://orcid.org/0000-0003-2790-5080

References

Adam A (2006) *Artificial Knowing: Gender and the Thinking Machine*. New York: Routledge.

Agre P (2002) The practical logic of computer work. In: Scheutz M (ed.) *Computationalism: New Directions*. Cambridge, MA: The MIT Press, 129–141.

Agre P (2004) Internet research: For and against. In: Consalvo M, Baym N, Hunsiger J, et al. (eds) *Internet Research Annual*, vol. 1. New York: Peter Lang, 25–36.

Albrechtsen H (2015) This is not domain analysis. *Knowledge Organization* 42(8): 557–561.

Anderson C (2008) The end of theory: The data deluge makes the scientific method obsolete. *Wired*, 16 July. Available at: https://www.wired.com/2008/06/pb-theory/ (accessed 17 April 2019).

Arango G (1989) Domain analysis: From art form to engineering discipline. *ACM SIGSOFT Software Engineering Notes* 14(3): 152–159.

Atkins DE, Droegemeier KK, Feldman SI, et al. (2003) Revolutionizing science and engineering through Cyberinfrastructure. Report of the Blue-Ribbon Advisory Panel on Cyberinfrastructure, National Science Foundation, Alexandria, VA.

Berg M (1997) *Rationalizing Medical Work: Decision-Support Techniques and Medical Practices*. Cambridge, MA: The MIT Press.

Boden MA (2008) *Mind as Machine: A History of Cognitive Science*. Oxford: Oxford University Press.

Buchanan B and Shortliffe EH (1984) *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading: Addison-Wesley.

Burrell J (2016) How the machine 'thinks': Understanding opacity in machine learning algorithms. *Big Data & Society* 3(1): 1–12.

Calvert J (2006) What's special about basic research? *Science, Technology, & Human Values* 31(2): 199–220.

Collins HM (1990) *Artificial Experts: Social Knowledge and Intelligent Machines*. Cambridge, MA: The MIT Press.

Daston L and Galison PL (1992) The image of objectivity. *Representations* 40: 81–128.

Davis R, Buchanan B and Shortliffe E (1977) Production rules as a representation for a knowledge-based consultation program. In: Reggia JA and Tuhrim S (eds) *Computer-Assisted Medical Decision Making* (Computers and Medicine). New York: Springer, 3–37.

Dreyfus HL (2007) Why Heideggerian AI failed and how fixing it would require making it more Heideggerian. *Philosophical Psychology* 20(2): 247–268.

Ensmenger NL (2012) *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise*. Cambridge, MA: The MIT Press.

Feigenbaum EA (1980) *Knowledge engineering: The applied side of artificial intelligence*. Stanford Heuristics programming project report no. HPP-80-21, Department of Computer science report no. STAN-CS-80-812. Stanford, CA: Stanford University.

Feigenbaum EA and Buchanan BG (1993) DENDRAL and meta-DENDRAL: Roots of knowledge systems and expert system applications. *Artificial Intelligence* 59(1): 233–240.

Forsythe DE and Buchanan BG (1989) Knowledge acquisition for expert systems: Some pitfalls and suggestions. *IEEE Transactions on Systems, Man, and Cybernetics* 19(3): 435–442.

Fujimura J and Chou DY (1994) Dissent in science: Styles of scientific practice and the controversy over the cause of AIDS. *Social Science & Medicine* 38(8): 1017–1036.

Gibbons M, Limoges C, Nowotny H, et al. (1994) *The New Production of Knowledge: The Dynamics of Science and Research in Contemporary Societies*. London: SAGE.

Gieryn TF (1983) Boundary work and the demarcation of science from non-science: Strains and interests in professional ideologies of scientists. *American Sociological Review* 48(6): 781–795.

Goody J (1986) *The Logic of Writing and the Organization of Society*. Cambridge: Cambridge University Press.

Hacking I (1992) 'Style' for historians and philosophers. *Studies in History and Philosophy of Science Part A* 23(1): 1–20.

Keating P and Cambrosio A (2012) *Cancer on Trial: Oncology as a New Style of Practice*. Chicago, IL: The University of Chicago Press.

Kroll JA, Barocas S, Felten EW, et al. (2016) Accountable algorithms. *University of Pennsylvania Law Review* 165: 633–705.

Latour B (1988) *The Pasteurization of France*. Cambridge, MA: Harvard University Press.

Latour B and Woolgar S (1979) *Laboratory Life: The Construction of Scientific Facts*. Beverly Hills, CA: SAGE.

Lisboa LB, Garcia VC, Lucrédio D, et al. (2010) A systematic review of domain analysis tools. *Information and Software Technology* 52(1): 1–13.

Lohr S (2018) M.I.T. plans college for artificial intelligence, backed by $1 billion. *The New York Times*, 15 October. Available at: https://www.nytimes.com/2018/10/15/technology/mit-college-artificial-intelligence.html (accessed 17 April 2019).

McCallum A, Nigam K, Rennie J, et al. (1999) A machine learning approach to building domain-specific search engines. In: *Proceedings of the international joint conference on artificial intelligence (IJCAI-99)*, Stockholm, 31 July–6 August, 662–667. San Francisco, CA: Morgan Kaufmann.

McCarthy J and Hayes PJ (1981 [1969]) Some philosophical problems from the standpoint of artificial intelligence. In: Webber BL and Nilsson NJ (eds) *Readings in Artificial Intelligence*. Los Altos, CA: Morgan Kaufmann, 431–450.

McCarthy MT (2017) The semantic web and its entanglements. *Science, Technology & Society* 22(1): 21–37.

Mol A (2008) *The Logic of Care: Health and the Problem of Patient Choice*. London: Routledge.

Monea A (2016) The graphing of difference: Numerical mediation and the case of Google's knowledge graph. *Cultural Studies ↔ Critical Methodologies* 16(5): 452–461.

Neighbors JM (1980) *Software construction using components*. PhD Thesis (Technical report TR-160). ICS Department, University of California, Irvine, CA.

Neighbors JM (1989) Draco: A method for engineering reusable software systems. In: Biggerstaff T and Perlis A (eds) *Software Reusability, Vol. 1: Concepts and Models*. Reading: Addison-Wesley, 295–319.

Newell A, Shaw JC and Simon HA (2000 [1959]) Report on a general problem solving program. In: Chrisley R (ed.) *Artificial Intelligence: Critical Concepts*, vol. II. London: Routledge, 69–87.

Nilsson NJ (2009) *The Quest for Artificial Intelligence*. New York: Cambridge University Press.

National Research Council (NRC) (1992) *Computing the Future: A Broader Agenda for Computer Science and Engineering*. Washington, DC: The National Academies Press.

National Science Foundation (NSF) (2016) Frequently Asked Questions (FAQs) for NSF 16-615: Transdisciplinary Research in Principles of Data Science Phase 1 (TRIPODS). Available at: https://www.nsf.gov/pubs/2017/nsf17048/nsf17048.jsp?WT.mc_id=USNSF_25andWT. mc_ev=click%20-%20q19#q19 (accessed 17 April 2019).

National Science Foundation (NSF) (2018) Partnerships between Science and Engineering Fields and the NSF TRIPODS Institutes: TRIPODS + X (NSF Program Solicitation 18-542). Available at: https://www.nsf.gov/pubs/2018/nsf18542/nsf18542.htm (accessed 17 April 2019).

Poirier L (2018) Making the Web meaningful: A history of Web semantics. In: Brügger N and Milligan I (eds) *The SAGE Handbook of Web History*. London: SAGE, 256–269.

Porter A and Rafols I (2009) Is science becoming more interdisciplinary? Measuring and mapping six research fields over time. *Scientometrics* 81(3): 719–745.

Prieto-Díaz R (1990) Domain analysis: An introduction. *ACM SIGSOFT Software Engineering Notes* 15(2): 47–54.

Rheinberger HJ (1992) Experiment, difference, and writing: I. Tracing protein synthesis. *Studies in History and Philosophy of Science Part A* 23(2): 305–331.

Ribes D (2018) STS, meet data science, once again. *Science, Technology, & Human Values* 44(3): 514–539.

Ribes D and Bowker GC (2009) Between meaning and machine: Learning to represent the knowledge of communities. *Information and Organization* 19(4): 199–217.

Ribes D and Finholt TA (2009) The long now of technology infrastructure: Articulating tensions in development. *Journal for the Association of Information Systems* 10(5): 375–398.

Schank RC and Jona MY (1994) Issues for psychology, AI, and education: A review of Newell's Unified Theories of Cognition. In: Bobrow DG (ed.) *Artificial Intelligence in Perspective*. Cambridge, MA: The MIT Press, 375–388.

Simon HA (1973) The structure of ill structured problems. *Artificial Intelligence* 4(3–4): 181–201.

Slezak P (1989) Scientific discovery by computer as empirical refutation of the strong programme. *Social Studies of Science* 19(4): 563–600.

Star SL (1989) The structure of ill-structured solutions: Boundary objects and heterogeneous distributed problem solving. In: Huhns M and Gasser L (eds) *Readings in Artificial Intelligence*. London: Pitman, 37–54.

Stengers I (2000) *The Invention of Modern Science*. Minneapolis, MN: University of Minnesota Press.

Suchman LA and Trigg RH (1993) Artificial intelligence as craftwork. In: Chaiklin S and Lave J (eds) *Understanding Practice: Perspectives on Activity and Context*. Cambridge: Cambridge University Press, 144–178.

Van Melle WJ (1980) *A domain-independent system that aids in constructing knowledge-based consultation programs*. PhD Thesis, Department of Computer Science, Stanford University, Stanford, CA.

Verran H (2001) *Science and an African Logic*. Chicago, IL: The University of Chicago Press.

Wand Y and Weber R (1993) On the ontological expressiveness of information systems analysis and design grammars. *Information Systems Journal* 3(4): 217–237.

Winograd T (1975) Frame representations and the declarative/procedural controversy. In: Bobrow D and Collins A (eds) *Representation and Understanding: Studies in Cognitive Science*. New York: Academic Press, 185–210.

Author biographies

David Ribes is associate professor in the Department of Human Centered Design and Engineering (HCDE) and director of the Data Ecologies Lab (deLAB) at the University of Washington. He is a sociologist of science and technology.

Andrew S Hoffman is an ethnographer of data infrastructures, working at the intersections of STS and pragmatic sociology. He is currently a Research Scientist in the Data Ecologies Lab (HCDE, UW) and in Summer 2019 will be joining the iHub for Security, Privacy, and Data Governance at Radboud University in the Netherlands.

Steven C Slota is a Post-Doctoral Researcher at the University of California, Irvine, and is currently a visiting scholar at the Data Ecologies Lab of the University of Washington department of Human Centered Design and Engineering (HCDE). His areas of research interest include infrastructure studies, information and data ethics, and the role of knowledge production in policy processes.

Geoffrey C Bowker is Chancellor's Professor and Donald Bren Professor in Information and Computer Sciences, Department of Informatics, University of California, Irvine. He has studied scientific Cyberinfrastructures since their inception.