Appears in proceedings of World Congress on Social Simulation (2008)

Playing 20 Questions in Social Science – toward "Social Science Architectures" ?

Glenn Taylor, Robert E. Wray, PhD, Randolph M. Jones, PhD

Soar Technology, Inc. 3600 Green Court, Suite 600
Ann Arbor, MI 48105
{glenn,wray,rjones}@soartech.com

Abstract. One result of the "cognitive revolution" was the development of cognitive architectures that incorporate cognitive theories into a computational form that can be used to inform and constrain cognitive model development. This paper attempts to draw lessons learned from cognitive architecture development as a potential roadmap for how to develop theory-based computational architectures for computational social science research.

Keywords: computational social science, cognitive architecture

1 Introduction

In the heyday of the "cognitive revolution," Allen Newell asserted that "You Can't Play 20 Questions with Nature and Win" [1] – that asking individual questions about individual cognitive phenomena could not lead to the formation of a "unified theory of cognition" – one that integrates a range of theories in a coherent, consistent manner, and which can be used to explain within that integrated theory the full range of cognitive phenomena. One result of this mindset, as combined with concepts from artificial intelligence, was the development of the concept of *computational cognitive architecture* that incorporates integrated cognitive theories into a computational form that can be used to inform and constrain theory-based cognitive model development.

In computational social science, it seems there is a similar kind of junction, in which there are many computational models focused on individual theories and phenomena in social science – e.g., innovation diffusion theory [2],. There are also a number of toolkits for developing computational social science models (e.g., iThink, RePast). However, there does not seem to be the equivalent to computational *cognitive architecture* in computational social science. That is, there are no modeling toolkits that implement an integrated theory to account for the full range *social* behavior, nor do the computational toolkits that exist in social science help build theory-based models.

This paper describes some of the concepts in computational cognitive modeling and cognitive architecture, in an effort to chart a parallel path between the development of cognitive architectures and a possibly equivalent concept in computational social science.

2 Defining the Landscape of Computational Modeling

To begin, we want to define some terms first to help ground the discussion. We approach these terms specifically from a *cognitive modeling* perspective (our background). These definitions are not meant to be controversial, but in some cases, we make fine distinctions to draw direction connections later when talking about computational social science modeling.

Theory – an explanation of some phenomena that is testable/falsifiable, makes predictions, etc.

Model – an instantiation of a theory about one or more phenomena observed in a particular system under study. A model may (and almost probably does) make simplifying assumptions, and in doing so may become an abstraction.

Modeling Framework – a set of concepts about some phenomena ("conceptual framework"), and sometimes a set of tools or toolkits ("computational framework"), that support a range of models with those concepts. For example, rules-based languages like OPS-5 or JESS can be considered computational frameworks that provide some primitives building cognitive models.

Computational Cognitive Model – a model of human cognition, in computational form, typically focused on a particular cognitive competency or phenomena, such as learning; the computational model implements a *theory* of how the cognitive competency works under some assumptions. The model can often be used to make predictions for a particular task – e.g., how long it will take humans to perform a math problem. Computational models will often be built using a computational framework – e.g., a particular language that has relevant constructs, such as rules.

Computational Cognitive Modeling Architecture – a computational framework that explicitly incorporates theories of human cognition, and which allows for building cognitive models based on those theories, especially to the extent that a single architecture can be used to build *multiple models* consistent with the underlying theories. Includes theorized cognitive *representations* and *processes* necessary for cognition – these become the *fixed* constructs available to modelers which are used to guide theory-based model development. Example Cognitive Architectures include Soar [3] and ACT-R [4]. In the rest of this paper, when we use the shorthand term *cognitive architecture*, we will typically mean *computational cognitive modeling architecture.*

In the next section, we will expand on the concept of cognitive architecture, and then begin to draw explicit ties to computational social science.

3 Cognitive Architecture

Cognitive architecture [5], from a non-computational "conceptual framework" perspective, is an attempt to describe the structures and processes of cognition as a whole. [6] fits this within the concept of a *unified theory of cognition (UTC)* – instead of accounting for narrow, individual phenomena in cognition (e.g., Fitts Law [7]), a UTC accounts for the full range of cognitive capabilities: problem-solving, learning, emotion, etc. The idea of an integrated cognitive architecture, incorporating all the necessary functional components of cognition, falls naturally out of the idea of a UTC. Essentially, a single cognitive architecture should *account for all cognitive phenomena*. Functional components of a cognitive architecture are meant to be analogous to components in the "architecture of the mind" (not the brain necessarily, but the mind). Notionally, a cognitive architecture might consist of a short-term memory, a working memory, sensory systems, etc. (of course, depending on the integrated cognitive theory on which the architecture is based). However, regardless the particular functional modules of the architecture, a hallmark of cognitive architecture in general is the specification of *fixed constructs* (namely, representations and processes over those representations) that are constant across all cognitive tasks. For example, a fixed representation might be rules for long-term memory; a fixed process might be the algorithms for memory retrieval – these are constants across all models built within the architecture. These constructs are often explicitly supported in a *programming language* for building model. (Indeed, many cognitive architectures provide their own programming language – e.g., Soar, ACT-R – to further enable and encourage theory-based model development.). The things that are invariant – for example, the particular knowledge about how to perform a task – would vary across tasks. These variants effectively become the data over which the architecture operates.

There is a range of cognitive architectures, each incorporating different theoretical constructs, and therefore each providing a different set of fixed computational constructs that can be used to build cognitive models. However, cognitive architectures almost by definition are meant to provide an integrated, theory-based framework for explaining human cognition. The fixed representations and processes become implementations of the theory, and furthermore impose *constraints* on model building – cognitive models built within a cognitive architecture *must* abide by those fixed constructs to be considered theory-based. This is not to say that cognitive architectures are all complete – they are in many ways idealizations of a complete theory of cognition. But they are also a sandbox to allow for theorists to try to integrate theories of cognition into a form that can be experimented with, that can be used to build models to generate predictions, etc. Cognitive architectures have in many ways enabled further theorizing about the nature of cognition by providing evidence for what functional areas must exist in the mind to be able to work as an integrated whole.

These core concepts of *accounting for all phenomena; fixed constructs; constraints; and programming language* will be revisited later in the context of computational social science.

3 Relating to Computational Social Science

Where cognitive science is about understanding and explaining observable phenomena in cognition, social science is focused on the observable phenomena of social systems. Obviously, social science is a very wide, multi-disciplinary field. Sociology might focus on group action; Political science might focus on voting patterns under some conditions; anthropology might focus on how culture changes over time; etc. Many of these fields might study the decentralized nature of behavior, and the "emergent" qualities of behavior from the interactions of multiple actors over time. These phenomena are often explained in the form of theories that can be used to make predictions about those social systems under different conditions.

Like cognitive science models, many social science models start with some phenomenon and a theory about that phenomenon. The model is an implementation of some theories in some form which is meant to generate predictions that match the original observed phenomenon in some way. By extension, *computational social science* is concerned with using a range of computational methods for building computer-based models that can be experimented with, used for prediction, etc., possibly leading to the refinement of existing theories or development new theories. There are thousands of computational social science models, typically focused on a single observable phenomenon in social science -- for example, the El Farol problem. These computational models as implementations of a theory make predictions by generating output that can be compared against data.

As with cognitive frameworks, there are analogs in social science. For example, systems dynamics modeling is a "conceptual framework" with primitives of stocks and flows (among other things) that allow models to be built using certain terms, and there exists computational analogs such as iThink that can be used to build computational systems dynamics models. Agent-based modeling is a "conceptual framework" that allows models to be built using different terms, and there are tools or toolkits for building computational agent-based models. (When talking about computational frameworks like iThink or RePast, we will use the shorthand term "toolkit.")

Very general computational toolkits exist for building social science models, for example MSExcel™ or Mathematica™. However, the primitives in these "toolkits" – that is, all of mathematics – are not usually very helpful in focusing a social scientist on the modeling problem at hand. One goal of a framework (conceptual or computational) is that the primitives offered by the framework match closely to the concepts used by the theorists. If a theorist casts their social phenomena in a certain way (e.g., agent-based), he or she might be drawn to these different kinds of modeling paradigms or toolkits. The closer these primitives in the framework are to how the end-users think about the problem, the easier it will be for those users to build models. Systems Dynamics modeling tools like iThink or STELLA offer primitives in the realm of systems dynamics – e.g., stocks and flows. Agent-based modeling is another paradigm for describing social science phenomena. A number of agent-based modeling frameworks exist: RePast, NetLogo, MASON, Ascape, StarLogo, Artisoc, etc. This means also that there are different abstractions for different kinds of users and problems. One role of a modeling framework is to define the "right" level of abstraction to enable models to be built easily, encapsulating many of the details of

how things work in such a way that modelers can focus on their specific modeling problem.

Focusing on agent-based modeling frameworks, these tend generally have two kinds of primitives: agents and interactions. These concepts are fairly broadly defined in these different frameworks. An *agent* can be any number of things, with any number of capacities. Often an agent is simply an abstract object (in the object oriented programming sense) that must be refined and instantiated by a modeler by writing, for example, Java code. Most agents perceive objects or interactions in their environment, and take some action in the environment, but these exact actions and perceptions often differ in every framework and every model. The conceptualization of an *interaction* can also vary widely, which may include simply observing one's neighbor, to (intentionally or unintentionally) leaving some signal in the environment, deliberately voting, etc. Interactions may change agent behavior, which may in turn change the interactions. This feedback loop is one of the hallmarks of agent-based models, and, indeed, social systems.

Overall, the "theory" that social systems consist of agents and interactions among those agents is not a terribly strong theory, in that by itself it cannot make strong predictions about those social systems. In this way, we claim that most if not all of the agent-based modeling frameworks described above are *theory free.* This is not a criticism as it might sound – that they are theory-free allows social science theorists working in an "agent-based" paradigm to build models that are themselves a rendition of a theory about a social system. However, one consequence of the generality of most of these frameworks is that the primitives they define are *underconstraining.* That is, the primitives in these frameworks do not help inform a social scientist modeler as to how to construct a theory-based model – except in that the model will have agents and interactions of some kind. It this lacking that leads us to think about the possibility for theory-based frameworks – *computational social science architectures* – that can help social scientists develop theory-based models.

4 Toward a Computational Social Science Architecture

Overall computational cognitive architectures have had a positive impact on cognitive science, in terms of helping to push the idea of integrated conceptual models, helping to understand what functional areas are required for integrated cognition, and even practical matters of making the building of computational cognitive models easier. Therefore, it seems natural to extend this metaphor to social science.

What might a computational social science *architecture* (CSSA) look like? Here we describe a range of possible characteristics, inspired by the discussion of cognitive architectures above, as they might apply to a CSSA. These are only a few such characteristics.

4.1 One architecture, multiple models

One idea behind cognitive architecture is to help enable the unification of multiple theories of cognition into a coherent, consistent whole that allows for the modeling of a range of cognitive phenomena within a single complex task. For example – not just a cognitive model of learning items in a list, or recall of items from that list, but learning and recalling while performing an algebra task. The complex task requires a range of cognitive capabilities, and a model of that task in a cognitive architecture exhibits all those capabilities, making relevant predictions along the way of the system as a whole.

Similarly, a social system does not simply exhibit voting behavior alone, nor does not simply demonstrate the emergence of crowd behavior for visiting a bar, but instead the complex "task" of a real society encompasses all of these things over a long period of time, and these sub-tasks themselves interact in potentially interesting ways. An integrated social theory should capture these simultaneously, and a computational framework should incorporate the constructs related to the integrated theory to enable building social science models that exhibit a range of phenomena.

4.2 Fixed Representations and Processes

One distinguishing characteristic of cognitive architecture is in the definition of *fixed representations and processe*s that embody the theories in the architecture, and which constrain model development to be consistent with the theories. Some similar concepts appear in existing social science modeling frameworks, as mentioned earlier. For example, agent-based modeling toolkits define *agents* and *interactions* as the primitive constructs. However, in these existing frameworks, these concepts do not really constrain model development because they are so generically defined. A CSSA that is grounded in theory would likely constrain the characteristics of those agent objects – what they can perceive, how they perceive, how they process information (not unlike Cognitive Architectures, perhaps). It would also constrain how they can interact with other agents and their environments. Further, a CSSA would rigorously define the nature of those interactions, what constitutes an interaction, etc. This is not to say that there should be a fixed "emergence" construct in a CSSA (this would perhaps contradict the non-centralized notion of emergence in the first place), but where theories of emergent behavior make strong predictions, the *constraints* informed those theories should be somehow incorporated into the definitions of how interaction takes place.

4.2 Programming Language Constructs

Often, cognitive architectures come with a programming language that contains the primitives represented in the theory as first-class objects or methods in the language – e.g., goals, operators, plans, problem solving methods, etc. This tends to be the case in existing agent-based modeling frameworks, where there is an *agent* object, for example, that might have some characteristics (e.g., unique identifier, location) and

methods related to its environment (e.g., move) or related to interactions with other agents (e.g., inform, vote). However, where these characteristics or methods are based in theory, it is up to the social scientist to define what they are and write them from scratch into each model. An appropriate architecture would define a reusable set of these primitives at the right level of detail for building a large class of models, and make these part of the language – both to ease the process of building models, but also to encourage the development of theory-based models by abiding by the constraints naturally in the language. If the language is too general or otherwise lacks the right kinds of primitives, modelers must invent their own primitives, and by doing so may are not necessarily constrained by the theory embedded in the architecture.

5 Conclusions

In this paper, we have described computational cognitive architectures and have attempted to draw parallels to computational social science. In this, we have suggested the possibility of developing a theory-based computational social science architecture, and have suggested some characteristics of such an architecture as inspired by cognitive architectures.

Given the huge space that is social science, it is perhaps unlikely that there is a single unifying computational social science architecture. However, maybe within a single discipline or range of theories, this is more feasible in the short term. Where there is coalescence of micro theories into unified theories, this is where computational social science architecture might be possible, and which might make it easier for social scientists to develop theory-based computational models and experiments that can more easily be shared, compared, repeated, validated, etc. Where cognitive architecture has helped advance cognitive science by enabling the development of tangible models that help make predictions about human cognition, as well as advancing and refining theories of human cognition, perhaps the same can happen for the social sciences.

References

1. Newell, A. *You can't play 20 questions with nature and win*. in *Visual Information Processing*. 1973: Academic Press.
2. Rogers, E., *Diffusion of Innovations*. 5 ed. 2003: Free Press.
3. Laird, J.E., A. Newell, and P.S. Rosenbloom, *Soar: An architecture for general intelligence.* Artificial Intelligence, 1987. **33**(3): p. 1-64.
4. Anderson, J. and C. Lebiere, *The Atomic Components of Thought*. 1998: Lawrence Erlbaum.
5. Pylylshyn, Z., *The Role of Cognitive Architecture in Theories of Cognition*, in *Architectures for Intelligence*, K. Van Lehn, Editor. 1991, Lawrence Erlbaum Associates: Hillsdale, NJ.
6. Newell, A., *Unified Theories of Cognition*. 1990, Cambridge, MA: Harvard University Press.

7. Fitts, P.M. and R.E. Jones, *Analysis of Factors Contributing to 460 "Pilot-Error" Experiences in Operating Aircraft Controls*. 1947, Aero Medical Laboratory, Air Materiel Command, Wright Patterson Air Force Base: Dayton, Ohio.