A Canonical Agent Model for Healthcare Applications

John Fox, David Glasspool, and Sanjay Modgil, *Advanced Computation Laboratory, Cancer Research UK*

A general autonomous agent system architecture has emerged from the healthcare application domain. Twelve invariant input-output patterns, or signatures, summarize its component properties, and shared data structures define the component interactions.

Although the utility of autonomous agents and multiagent systems in healthcare applications is now well established, agent technologies themselves remain somewhat immature and, from a theoretical point of view, often ad hoc. We've developed a standard, or canonical, agent model that's intended to be both theoretically well motivated and technically well defined while permitting alternative instantiations.

Our starting point is the domino agent model.[1] The Advanced Computation Laboratory developed this model for healthcare applications, but its design supports general-purpose cognitive agents. The domino model is similar to the classical beliefs-desires-intentions framework, but goes beyond BDI by defining a complete set of processes to transform between mental states, including a flexible decision-making framework based on logical argumentation.

The domino model reflects current trends in software agent design, but it has broader justification in its embodiment of features common to many disciplines and theories of cognitive systems, including neuroscience and cognitive psychology as well as AI.[2] For example, a number of basic cognitive functions are widely held to be required by any intelligent agent: perception and interpretation of the agent's environment, goal setting and maintenance, problem solving, decision making, plan assembly, plan execution, and action selection. There is also a consensus on the types of representation on which these processes operate: beliefs, goals, and plans are assumed in a wide range of approaches. In general terms, these functions and representations are present across many theoretical approaches to cognitive agents.[3–5]

The domino agent also provided the foundation for a practical agent language, ProForma (www.openclinical.org/gmm_proforma.html).[6] ProForma has been used extensively to construct healthcare applications such as decision support and clinical-workflow management.

The domino model has been significantly extended within the Argumentation Services Platform with Integrated Components project. Funded by the European Commission, ASPIC involves a broad consortium of partners concerned with the uses of argumentation in agent systems, including nonmonotonic reasoning, decision making, interagent dialogue, and learning. Our canonical model captures the extended model in a general, implementation-independent way that provides a practical foundation for specific system implementations and agent-implementation tools. To address the need for canonical abstraction, we've adopted software engineering's concept of signatures—a technique for defining software patterns or invariant procedural properties.

A general model

Healthcare applications have a number of additional requirements beyond the basic functions and representations that are common to many cognitive-system theories. (The "Related Work in Multiagent Healthcare Systems" sidebar describes four multiagent healthcare systems that, in different ways, illustrate these requirements.) On the basis of our experience with healthcare systems, we've identified three key requirements over and above the basic domino model:

- A communication capability for interactions between agents, which is important for multiagent

1541-1672/06/$20.00 © 2006 IEEE
Published by the IEEE Computer Society

Related Work in Multiagent Healthcare Systems

Jun Huang, Nicholas R. Jennings, and John Fox described an early multiagent system for a distributed healthcare application in which each agent could reason and make decisions.[1] They demonstrated their approach's technical adequacy in an experimental application for managing breast-cancer treatment. Each agent in the application could make decisions based on a standard technique. Huang's agents implemented part of the domino model,[2] (the left side of the model in figure 1 in the main article): reasoning about beliefs, raising goals, problem solving to identify possible solutions to the goals, and deciding which of the solutions was best in the circumstances. Agents could have different expertise and seek assistance from another agent in certain circumstances. For example, "general practitioner" agents might have broad knowledge of general medicine and a patient but need the deeper expertise of a "cancer specialist" agent on more difficult problems in that domain. The specialist could delegate part of its responsibility for certain clinical tasks to the general practitioner and through the general practitioner to specialist nurses. Delegation gave Huang's system a degree of robustness because if an agent acting for one of the healthcare professionals refused or dropped a commitment, the managing agents could reassign the required task.

Elizabeth Black also demonstrated the technical feasibility of an agent network in the breast cancer domain.[3] She implemented a number of proxy agents for supporting general practitioners and cancer specialists in clinical genetics. All the agents could follow guidelines and enact simple clinical plans. They could also make decisions (for example, decisions to refer patients from a primary care unit to a genetics clinic). The agent architecture was based on the domino model[2] and used the PROforma agent and process specification language to represent decision guidelines, plans, and other tasks.[4] Black used a similar but more specialized agent model to investigate several patient-referral strategies, in which one agent requests another to carry out a specialist service. She found significant potential benefits in speeding up the journey breast cancer patients make through a number of clinical processes.

The Carrel system is being developed in Spain's Catalonia region to support management of organ transplants.[5] The system uses an agent institution, a system that defines protocols and norms through which agents can argue the merits of particular allocations of tissues and organs to patients. Although Spain's National Transplant Service is among the world's best, it shares a problem common with all such services: many potentially available organs are wasted ("discarded"). Among the reasons for this wastage is that experts located at the donor's site have exclusive authority to assess the organ's viability for transplantation—a decision-making process that ignores the very real possibility of disagreements among experts on such matters. As well as helping identify potential recipients for a donated organ across multiple transplant units, the Carrel system will therefore also coordinate the joint deliberations of donor and recipient agents as they determine appropriate assignment decisions. The system carries out its joint deliberation through an argumentation-based dialogue, in which a mediator agent evaluates the exchanged arguments to determine the winning argument. As a result, an organ that would ordinarily be discarded because a donor agent considered it nonviable could be transplanted if a recipient agent successfully argues its viability for the particular recipient the agent represents.

Cancer Research UK is developing the CREDO project to support women with proven breast cancer and women who are well but at risk because of genetic predisposition. The CREDO model of the "patient journey" for breast cancer care includes some 220 different services, packaged into approximately 12 different specialist packages; each package can be implemented as a PROforma agent.[4] CREDO aims for a comprehensive management approach. In addition to delegating tasks and managing workflows, reminiscent of Huang's approach, CREDO will capture data, recognize and interpret clinical events, notify colleagues of adverse or other significant circumstances, modify plans on the fly, and so on. This degree of flexibility and versatility involves significant technical challenges, including effective communication and coordination, prompt and safe decision making, collaborative treatment planning and timely execution of plans, and the ability to avoid repeating mistakes.

References

1. J. Huang, N. Jennings, and J. Fox, "An Agent-Based Approach to Health Care Management," *Int'l J. Applied Artificial Intelligence*, vol. 9, no. 4, 1995, pp. 410–420.
2. S. Das et al., "A Flexible Architecture for Autonomous Agents," *J. Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 4, 1997, pp. 407–440.
3. E. Black, "Using Agent Technology to Model Complex Medical Organizations," *AgentLink News*, Apr. 2003, pp. 7–8.
4. J. Fox et al., "Decision Support for Healthcare: The PROforma Evidence Base," *Informatics in Primary Care*, vol. 14, no. 1, 2006, pp. 49–54.
5. P. Tolchinsky, S. Modgil, and U. Cortes, "Argument Schemes and Critical Questions for Heterogeneous Agents to Argue over the Viability of a Human Organ," *Papers from AAAI Spring Symp.*, tech. report SS-06-01, AAAI Press, 2006; http://www.aaai.org/Library/Symposia/Spring/ss06-01.php.

systems but not supported by the formalisms proposed for modeling clinical guidelines, workflows, and so on.

- A well-developed model of decision making under uncertainty, which is generally regarded as fundamental to dealing with the complexities of clinical practice. Researchers have described how to embed this capability in an agent system and incorporate logical argumentation techniques for decision making.[1,6,7]
- The ability to access or communicate the knowledge and arguments used in specific decisions, a requirement that supports collaborative decision-making in multiagent applications.

We extended the domino model to meet these requirements. Figure 1 shows the extended model. It's built around the six basic entities of the original domino model, which can

itself be seen as an extension of the BDI agent model. In our terminology, beliefs are aspects of the agent's environment or its own mental states, which the agent holds to be true (that is, the agent will act upon them while they continue to hold). Goals are equated with "desires" and plans with "intentions." We view intentions as commitments to new beliefs or to carrying out certain plans or pursuing new goals in the future.

A fundamental capability of the agent model is the ability to make decisions under uncertainty—that is, to make choices between competing beliefs or alternative plans given a lack of certain knowledge about the true state of the environment or about the consequences of possible actions on the environment. The model introduces a four-step decision procedure in which an agent can identify decision options (competing beliefs or plans), construct arguments for and against the options, assess the relative strength of the sets of arguments for alternative options, and commit to the most-preferred option.[1,6]

The decision procedure reflects our primary ASPIC project activity—namely, to develop an agent framework that can integrate the different roles of argumentation in a principled way. Two features of the extended model accommodate this activity:

- *Interagent dialogue models*. Project partners are developing and formalizing interagent dialogue models (www.argumentation.org), and we're incorporating the results into our extended model for use in the Carrel and CREDO applications described in the sidebar. For example, we're extending standard FIPA-like performatives to include those that facilitate coordination on collaborative tasks, such as joint decision making or service negotiation, where deliberative or dialectical argumentation between agents is required.
- *Machine learning*. Project partners are investigating the relationship between argumentation and machine learning. Learning capabilities are especially important in healthcare applications, because human errors and system failures will occasionally occur no matter how well we design our systems. To support learning from experience and corrections to procedures, the agent platform should on all occasions maintain records of what happened, what decisions were taken and why, and what the outcomes were.

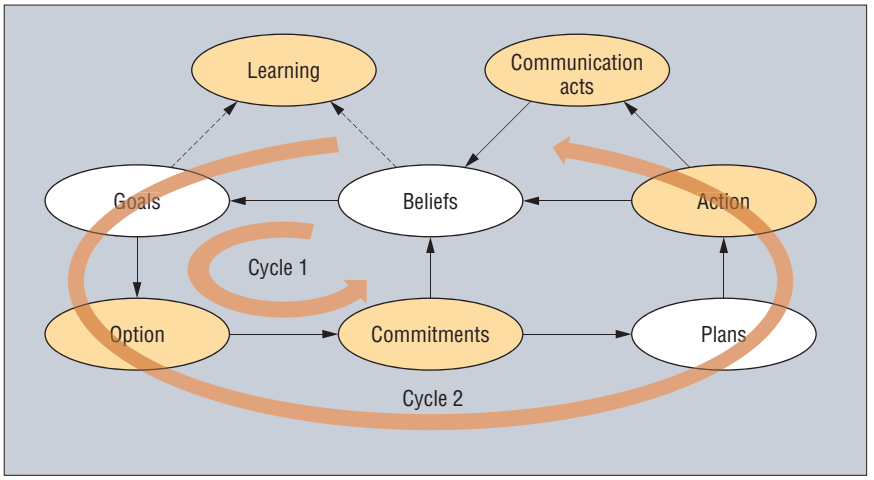


Figure 1. A general agent model subsuming the classic beliefs-desires-intentions framework (white ellipses, where goals are equated with desires and plans with intentions). Cycles 1 and 2 refer to the first two processing cycles of the example healthcare scenario. The dotted lines to the Learning entity indicate that there are other options for linking learning into the agent model.

From agent model to agent design

We now aim to formalize the agent model in a way that captures the primary functions shown in figure 1 in a general and implementation-independent way, yet which can provide practical foundations for specific agent design and implementation tools. To address this we have adopted the concept of a signature from software engineering which is a technique for defining software "patterns" or invariant properties of software procedures.

The signatures abstractly represent the pre- and post-conditions of a procedure. Figure 2 presents our proposal for a set of signatures covering the central functions of a "canonical" agent[1–8] augmented with signatures for dialogue and learning.[9–12] The ASPIC project's formal inference model for argumentation has informed the model's development, as have the project's use of arguments in decision making and dialogue, and the integration of argumentation with learning.

The signature variables are complex types. For example, the signature T1 abstractly describes a procedure for making inferences over beliefs. Consider the informal belief, "Mrs. Smith is at risk of a heart attack." We could model this belief as a pair [Context, Claim] in which Context is "Mrs. Smith" and Claim is atRiskOf("heart attack"). In each signature, Theory is a complex type representing domain knowledge or anything that we can formalize as a collection of predicates (for example, a logic program). The intuitive interpretation of this signature is simply that, given a set of beliefs about a particular situation, the agent can infer all beliefs that logically follow from the agent's current beliefs and knowledge. At this level of abstraction, the signature subsumes many different specific forms of inference, including monotonic and nonmonotonic logical inference and probabilistic or other quantitative models.

In a more sophisticated agent, the signature might remain the same but the belief's type is more complex. For example, consider "Mrs. Smith is believed to be at risk of heart attack because of an abnormal ECG." We could follow Stephen Toulmin's classic approach to modeling argumentation,[12] modeling this as a triple [Context, Claim, Warrant] in which the third-place Warrant represents a justification for Claim in this Context. In this scheme, we can distinguish distinct lines of reasoning (for example, "Mrs. Smith is at risk of a heart attack because she had one last year") because the two warrants ("abnormal ECG" and "previous heart attack") are explicit. When we represent the belief as a pair, the warrant remains implicit in the theory and the distinction between the different lines of reasoning isn't directly represented in the agent's mental state.

In our proposal, we go a step further, modeling an agent's beliefs as 4-tuple type definitions:

$$\text{Belief} \rightarrow_{\text{def}} (\text{Context, Claim, Warrant, Qualifier}) \qquad \text{(T1)}$$

S1. Inference Belief × Theory Belief	**S2. Goals** Belief x Theory OR Goal × Theory Goal	**S3. Problem solving** Goal x Belief x Theory Option
S4. Construct arguments Goal x Option × Belief × Theory OR Arg x Option × Belief × Theory Arg	**S5. Evaluate arguments** Goal × Option x Arg × Theory Merit	**S6. Commit to option** Goal x Option x Merit x Theory Belief OR Plan
S7. Plan enactment Goal x Plan x Theory Act OR Plan	**S8. Action execution** Act × Resource Action OR Message	**S9. Communication** Message × Theory Belief OR Theory
S10. Learning: MLBA Belief × Goal × Theory Theory	**S11. Learning: ABML** Belief × Arg × Theory Theory	**S12. Learning: CBL** Belief × Option Theory

Figure 2. Signatures for the canonical agent model's primary functions. In each signature, the variables above the line represent an input data pattern that instantiates each variable with a 4-tuple of the appropriate form for the type. The variable below the line represents the component's output and is similarly instantiated.

where Qualifier (a term also adopted from Toulmin) makes the relationship between Warrant and Claim explicit. For example, the qualifier's value might be strict, indicating that Claim is asserted to be deductively (monotonically) true, or defeasible, indicating that Claim holds until such time as a successful attack is mounted against its Warrant. In the latter case, Qualifier can also represent an argument's defeated status.

This general signature and type structure can also accommodate degrees of belief—for example, "Mrs. Smith is believed to be at risk of heart attack with certainty 0.5 because of an abnormal ECG." There are many schemes for quantifying uncertainty levels in inference and belief maintenance: Bayesian or objective probabilistic inference, possibility theory, certainty factors, and so on. We can model many such schemes as instances of this general signature.

In the remainder of this section, we'll summarize those signatures in figure 2 that are associated with various kinds of argumentation. Type definitions T1–T6 define the principal types for the left side of the agent model in figure 1, used in signatures S1–S6. In each case, we've adopted a 4-tuple type similar to that for beliefs, with a context in the first place and the primary content in place 2. The model supports variable types at different complexity levels; lower-order (place 2) types have lower computational demands, and higher types (places 3 and 4) give more flexibility but at greater computational cost.[8] We've elected to adopt more complex types throughout the canonical model, as this facilitates nonmonotonic computation (through place 3) and more reflective (self-aware) mental states through place 4. We reserve place 3 for the context and other information that justifies the primary content; place 4 is for an appropriate modality (a generalization of Toulmin's "qualifier").

Goals

It is a truism of AI that to build autonomous agents that can cope with complex, unpredictable environments, we must separate the cognitive and rational aspects of intelligence (what the agent believes about a particular situation and how it decides to act) from the operational aspects (what it actually does). The key concept here is the agent's goal, which determines what the agent wishes to achieve, not how it will achieve it. In BDI terms, the agent's desires link its beliefs to its intentions. Drawing on the discussion by Michael Winikoff and his colleagues,[13] we can define a set of properties characterizing a rational agent's goals. The goals must be

- explicit (if an agent is to reason about them),
- pertinent (with respect to some initiating situation or challenge),
- consistent (goals should not conflict), and
- persistent (goals exist so long as their success or failure conditions are not satisfied).

A type definition for Goal in signatures S2–S7 and S10 of figure 2 is

$$\text{Goal} \rightarrow_{\text{def}} (\text{Context, Objective, Issue, Status}) \quad \text{(T2)}$$

where Objective is a term representing the state that the agent wishes to bring about in the Context, such as a threat or an opportunity, and Issue represents the circumstances that led to the goal being raised (Issue ≠ Context). Drawing on Winikoff's discussion again, a goal can have two other properties—namely, whether it is achievable and whether it is achieved. The signature's place 4 is required for Status—that is, information about whether the goal is currently active (that is, pertinent, achievable, and not yet achieved) or dropped (achieved or not pertinent or not achievable).

Options

Goals can be epistemic (for example, an agent has a goal to interpret, explain, or predict events in the world) or practical (for example, the agent wishes to prevent or bring about situations in the world that are consistent with its desires). Whatever type of goal the agent has, it must find a solution to the challenge that initiated it. Because multiple viable solutions might exist, we've adopted the term Option for a viable candidate. Its type definition is

$$\text{Option} \rightarrow_{\text{def}} (\text{Context, Candidate, Rationale, Viability}) \quad \text{(T3)}$$

where Candidate can be a simple term, such as an atom naming the option, or—in more interesting situations—a complex object. For example, if the goal is to explain an observed event, an option might be an extended narrative or causal graph. If the goal is to achieve some change in the world, Candidate might be the result of a planning or design process. Rationale is the justification for Candidate, and Viability is Candidate's current status. An option can be viewed as viable while the status of an explanation for some event is convincing or a plan's preconditions remain satisfied, but otherwise it's nonviable. Whatever the type of thing that Candidate is intended to be, signa-

ture S3 of figure 2 and its associated option type is intended to capture an invariant pattern shared by many different kinds of problem-solving capabilities that intelligent agents need. At first glance, it might appear that signature S3 can't be sufficient to cover a wide-enough range of cases for practical applications, but to date we haven't identified an example that we couldn't reduce to this basic pattern.

Decision making

When an agent has multiple options, it must usually choose among them. In our model, the choice is arrived at through an argumentation process.[1,6] The process involves several steps (signatures S4–S6, figure 2). After the agent identifies the decision options, it constructs arguments about each one, aggregates the arguments to yield a preference ordering, and makes a commitment to the preferred option. Each function has its own signature with characteristic pre- and postconditions.

The decision-making argument's type has a form similar to the type for the inference-making argument in T1:

$$\text{Argument} \rightarrow_{\text{def}} (\text{Context, Candidate, Warrant, Qualifier}) \quad \text{(T4)}$$

However, basic inferencing restricts Qualifier to represent whether the argument is accepted or defeated, whereas Qualifier can have additional properties in decision making, such as a polarity to indicate whether an argument is for or against Candidate.

Our decision model allows any number of arguments to support or oppose an option, so another step in the decision procedure is to establish the overall Merit of the options by weighing the pros and cons of their arguments. The simplest representation of Merit is a numerical value, which an agent can use to induce a preference order over the options. However, an ordinal or "logical" scale is also possible, based on increasingly stringent acceptability criteria according to the pattern of arguments about the options. Type definition T5 is intended to accommodate both qualitative and quantitative interpretations of Merit:

$$\text{Merit} \rightarrow_{\text{def}} (\text{Context, Candidate, Goal, Support}) \quad \text{(T5)}$$

Knowing that an option is the most preferred among a set of options might be sufficient for an agent to commit to it at a given point, but this approach won't always be satisfactory because an agent that's interacting with a dynamically changing world can receive new information at any time. Potentially, therefore, the agent can construct additional arguments for or against an option and defeat existing arguments at any time. Consequently, the preference order over the options could change. This situation calls for a principled stopping rule that lets the agent safely commit to an option in the knowledge either that nothing will change the current preference order or, more pragmatically, that it's better to commit now than to risk incurring greater costs through delay. The problem of stopping rules is discussed at greater length elsewhere.[1] A further type is required to represent an agent's commitments. This makes explicit both the candidate's commitment status (for example, "open," "defeasibly committed," "strictly committed") and the stopping rule that the agent has used:

An agent that's interacting with a dynamically changing world can potentially construct additional arguments for or against an option and defeat existing arguments at any time.

$$\text{Commitment} \rightarrow_{\text{def}} (\text{Context, Candidate, StopRule, CommitStatus}) \quad \text{(T6)}$$

In healthcare applications, a human expert (such as a clinician) must sometimes make the final commitment to a decision. In these cases, the computer system is restricted to offering a reasoned recommendation rather than taking the decision autonomously. Stopping rules can be domain-independent or domain-specific;[6] they are therefore defined within the Theory part of signature S6 (figure 2).

Plan enactment and action execution

An agent will typically pursue its goals via extended plans of action. It might construct plans specifically in response to a problem or select them from a library of precompiled plans appropriate to situations in which it typically finds itself. In either case, we treat the construction or selection of a suitable plan under signature S3, which treats an entire plan as an Option. The AI literature views a plan broadly: it comprises either a sequence or collection of more or less well-defined tasks (including specific actions, decisions, or subplans) or a set of subgoals to be achieved, leaving the details of how to do so to the circumstances that prevail when the plan is executed. Signature S7 is defined in terms that encompass all of these possibilities. The new types we encounter in S7 are Plan and Act. We define the Plan type as follows:

$$\text{Plan} \rightarrow_{\text{def}} (\text{Context, Schema, Rationale, Status}) \quad \text{(T7)}$$

where Schema is a relatively neutral term for the sequence of actions, subgoals, or subplans constituting the plan's content. T7 also represents the Rationale for the plan choice, and Status tracks whether a plan is active, discarded, successfully completed, or aborted—a set of states drawn from our experience in healthcare plan representation. This is the minimum set of states we have found necessary in that domain, though more complex schemes have also been used.[6]

At some point, a plan's enactment will "bottom out" in the execution of actions in the world. We define the Act type in signature S7 as follows:

$$\text{Act} \rightarrow_{\text{def}} (\text{Context, Action, Goal, Status}) \quad \text{(T8)}$$

where Action defines an action in the world in an implementation-specific way. The Act type includes the action's Goal, which the agent must know to monitor the action's success. The Status field can include the states "requested" (indicating that the agent has requested an action but has received no feedback on its outcome), "successful," and "failed" (indicating that the requested action has been taken with observable effects relative to its goal). Again, this is a practical minimum, but data requests in real clinical-information systems often use other domain-specific or application-specific status terms. Signature S8 governs the execution of actions. Taking an action in the world requires specifying both the action itself (the Act type) and the resource to be used to implement the action (example resources

might include a physical device or an external software system). The Resource field of S8 abstracts from both of these cases.

Communication and dialogue

Signature S8 indicates that one way to execute an action in the world is to send a message to another agent. In principle, this could be another artificial agent or a human. A message is a sufficiently important action subclass to merit separate treatment. We therefore define a Message type as follows:

$$\text{Message} \rightarrow_{\text{def}} (\text{Context, SpeechAct, Provenance, Status}) \quad \text{(T9)}$$

where the SpeechAct field contains the message content. Because messages are typically sent as part of a dialogue, the agent must track their origin so we need to record the speech act's provenance—that is, whether it originates with "self" or another agent and, if the latter, which one.

As signature S9 indicates, an agent may also receive messages from other agents. Given a dialogue theory, which all participants in a conversation must share, we can interpret another agent's SpeechAct as providing us with information. In general, we can gain two types of information from a message. Most straightforwardly, we might gain a new belief about the world. However, because the message might originate from an agent that has its own theories with which to interpret its beliefs (comprising its domain or general knowledge), we might alternatively gain a new or updated theory. Signature S9 reflects these two possible types of message content.

Learning

The final three signatures of figure 2, S10–S12, concern learning. They represent variations on a common theme that views learning as a process in which the agent can update or generate its own theories as a result of experience. The experience on which learning is based will generally include beliefs, but its other components can depend on the type of learning that's taking place. We've been particularly interested in three learning variants within the ASPIC project:

- *Machine-learning-based argumentation* (MLBA, S10) is a modified version of inductive-logic programming; the type of theory learned is a new set of arguments that can be applied to future induction and decision making.
- *Argumentation-based machine learning* (ABML, S11) is a novel method in which the agent can guide the learning process by providing arguments that can hint at the salient features to learn about a situation.
- *Case-based learning* (CBL, S12) includes methods that might consider an entire option (see T3) as the material for the learning process.

We're interested in integrating these learning methods into the agent framework, rather than their individual details, and we're particularly interested in the question of whether it will prove possible to reduce S10–S12 to an even more general form.

Implementation

The canonical agent model has been implemented using the Cogent cognitive modeling package,[14] which layers a variety of tools for cognitive modeling and simulation experiments on top of the Prolog programming language. The model uses a central working memory, which carries information about the agent state (beliefs, goals, arguments, and so on). Elements in the working memory comply with the type definitions T1–T9, and the current demonstration system includes components that implement signatures S1–S9 to read and write these elements.

The model includes two knowledge repositories containing facts, rules, schemas, and so on, which form the signatures' Theory or knowledge element:

- A general knowledge repository, intended to be identical in all of a particular implementation's agents, includes, for example, facts and schemas to implement a particular dialogue theory.
- A domain knowledge repository specifies knowledge that is unique to a single agent.

In our demonstration system, all agents are canonical—that is, all computational components comply with the relevant signatures. In fact, the only differences between agents are the contents of the domain knowledge repository.

The signatures in figure 2 specify general constraints on the pre- and postconditions of each functional component of the agent model. In each signature, the variables above the line represent an input data pattern that instantiates each variable with a 4-tuple of the appropriate form for the type. The variable below the line represents the component's output and is similarly instantiated with an appropriate 4-tuple. At the signature level, each function is a black box that might be implemented in any number of ways—for example, in a conventional procedural algorithm, a set of production rules, a logic program, or hardware.

The present Cogent version implements the signatures as declarative if … then "meta-rules." Each rule constantly monitors the working memory's state to determine whether an instance of its signature precondition is satisfied; if so, the rule asserts into working memory a tuple of the type specified in the postcondition. If a rule's preconditions are satisfied, the rule updates working memory appropriately. These updates can themselves satisfy other components' preconditions, leading to further updates and so on. This approach is in line with the view that agents are usually reactive, responding dynamically to changing circumstances. At a higher level, however, sequences of rules implement deliberative processes, such as reasoning, goal-based decision making, and plan enactment.

The signatures of figure 2 define the input-output conditions of each process in the canonical agent. We implemented each signature directly as a Cogent rule that defines the triggering conditions and the input/output types. The actual process underlying the signature is implemented as a Cogent condition definition (essentially a Boolean condition expressed in Prolog) and an action which updates the working memory. To change the implementation, a system designer could substitute a different condition definition while retaining the same signature rule.

Evaluation scenario

Our current implementation is being validated on a standard multiagent healthcare scenario developed as part of the EU's ASPIC project. We outline the prototype implementation's operation here by describing a scenario involving two agents, a *cardiac agent* and a *guardian agent*. The CA and GA carry out a shared task and communicate via a message bus that they both can read and write messages to and from. The two agents are identical in every respect except in their domain-specific knowledge. CA has knowledge relevant to the diagnosis and management of heart problems. GA, on the other hand, has knowledge related to medical safety, including knowledge related to the safety of drugs used under a variety of conditions. The two agents engage in the cooperative management of a patient, Mrs. Smith, who is suf-

fering from a suspected heart attack. Figure 3 shows their dialogue in schematic form.

This scenario represents a realistic, if simplified, healthcare application. The Cogent agent model implementation carries out the scenario exactly as we detail it here.

Processing within each agent roughly follows the cycle of processes we've defined. Figure 1 shows the scenario's first two cycles, which we'll describe in detail; subsequent cycles are described in less detail, but they are similar.

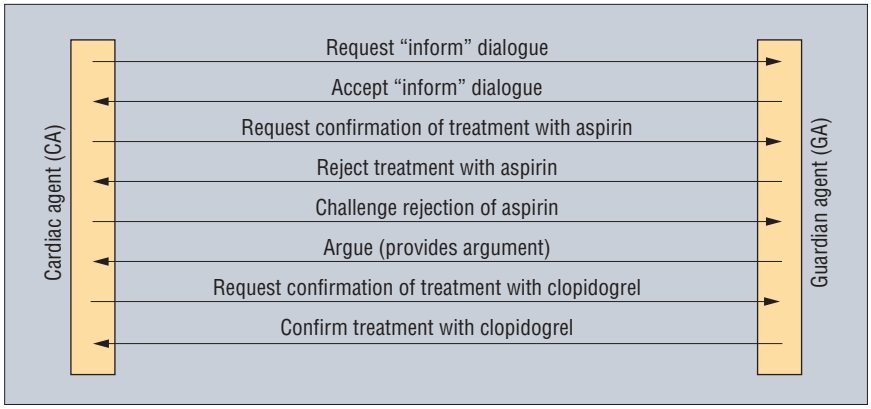


Figure 3. Outline of the dialogue between cardiac and guardian agents in the evaluation scenario. CA starts by proposing a drug (aspirin), which GA rejects as unsafe. CA challenges this rejection, and GA responds by providing the argument it used to reject the drug. CA incorporates this argument into its reasoning process and concludes that a different drug would be appropriate. GA confirms that this decision will be safe.

Cycle 1

CA starts with two beliefs: Mrs. Smith is elderly and she has chest pain. From its domain-specific knowledge about cardiac problems, the agent infers the possibility of a myocardial infarct (the inference process labeled Beliefs in figure 1, defined in signature S1 of figure 2) and raises a goal to manage this condition (the Goals process of figure 1 and S2). CA proposes two candidate solutions (S3) for the goal: treatment with the drugs clopridogrel or aspirin (this example is intentionally simplified, from a medical point of view).

The agent can construct several arguments (S4) for and against each candidate drug. Our prototype implementation constructs arguments based on effectiveness, cost, and availability because its domain knowledge contains the facts that clopridogrel and aspirin are both available and effective treatments, but only aspirin is cheap. The aggregation process (S5) determines that the balance of argument favors prescribing the cheaper generic drug aspirin. However, the commitment process (S6) determines from its domain knowledge (Theory) that CA isn't authorized to make a firm commitment to drug-prescribing decisions, so the commitment to use aspirin is qualified as "provisional."

Cycle 2

The presence of a provisional commitment now triggers the Goals process (S2) in CA to raise a goal to confirm the commitment. S3 can propose an option to achieve this goal by opening a "confirmation" dialogue with an appropriately authorized agent. CA's domain knowledge includes the fact that GA is authorized to confirm drug prescription decisions, and CA's S3 process introduces the intention to consult GA. The system is unable to raise any argument against this intention, so CA commits to it (S4–S6; CA isn't prohibited from carrying out dialogue actions and therefore makes a "firm" rather than "provisional" commitment). This triggers initiation of plan enactment (S7). The plan associated with the intention has two steps: request an "inform" dialogue with GA, then ask for confirmation of the prescription decision. CA executes the first of these steps (S7 and S8), making a dialogue move addressed to GA with type "request" (a request to open a dialogue) and content "inf" (an "inform" dialogue).

Remaining cycles

Referring to figure 1, the first two processing cycles within agent CA have involved most of the agent model's processing elements and have exercised signatures S1–S8. Cycle 1 traversed S1-S2-S3-S4-S5-S6, and cycle 2 traversed S2-S3-S4-S5-S6-S7-S8. Processing in the model generally follows this sequence: each metarule triggers its associated process immediately when its preconditions are met (which is usually, but not necessarily, the result of the preceding process's termination). The same general sequence of operation holds for both agents in the remainder of the scenario.

Referring to figure 3, GA receives the dialogue move (S9), leading to a belief that triggers a goal to respond to CA's dialogue request. A simple solution—to accept the request—passes unopposed through the argumentation and commitment phases and leads to GA executing a dialogue move accepting CA's dialogue request.

CA's plan-execution process sees the acceptance as a successful outcome of its first step and proceeds directly to the next step in its plan: a "question" dialogue requesting confirmation of the proposed administration of aspirin to Mrs. Smith.

GA responds by raising a goal to confirm CA's decision, and its S3 process proposes CA's intention to prescribe aspirin as an option for argumentation. GA's S4 process proceeds to construct arguments for and against aspirin treatment for Mrs. Smith. The arguments differ from the set raised by CA because GA has different domain knowledge. In fact, GA's S4 process raises a single argument based on a schema that essentially says, "If a treatment is proposed that is known to exacerbate a condition, and if the treated patient is known to be susceptible to that condition, then this constitutes an argument against the proposed treatment." GA has domain knowledge that aspirin exacerbates gastritis and that Mrs. Smith is susceptible to gastritis, so this argues against CA's provisional decision to prescribe aspirin.

GA can now satisfy its current goal by informing CA that its decision is rejected. Argumentation, commitment, and plan execution for this candidate plan take place in the usual way (but are, in fact, perfunctory), and GA makes a "reject" dialogue move.

CA recognizes the rejection as a failure mode for its most recent dialogue action. It marks that action "failed" and hence the plan that contained the action is "aborted." This triggers CA's S2 process to raise a goal to understand the reason for the rejection, a standard reaction defined in its general knowledge. One

option to achieve this goal is to carry out a "challenge" dialogue, eliciting an explanation from GA in the form of arguments that led to the rejection of aspirin. CA duly executes an appropriate dialogue action.

GA now raises a goal to respond to the challenge. Its S3 process constructs an "argue" dialogue move, containing the "gastritis argument" against aspirin.

CA receives the "argue" message containing the argument against aspirin and, consequently, revises its aggregation and commitments on the basis of the changed argument structure. Aspirin is now rejected, and clopridogrel accepted. Again, because the option "clopridogrel" represents a drug prescription plan, CA can only provisionally commit to it.

This time, however, GA can construct no arguments against CA's proposal to treat Mrs. Smith with clopidogrel, so it makes an "inform" dialogue move telling CA that its proposal is acceptable. CA can now convert its provisional intention to treat Mrs. Smith with clopridogrel to a firm commitment, and it begins enacting an appropriate prescription plan. CA requests that clopidogrel be administered to Mrs. Smith, which completes the plan, and so the execution finishes.

We intend the set of canonical signatures we've presented here to provide a clear, transparent definition of our model and a standard against which to compare alternatives and facilitate agent component reuse. It could also become the basis of a more sophisticated version of the ProForma language. We hope this article will kindle discussion about the utility of a standard language along these lines in healthcare. The model provides a practical framework for designing and realizing multiagent systems, but we believe its formalization in terms of canonical signatures could lead to insights into functions that are common to many agent theories and technologies. ■

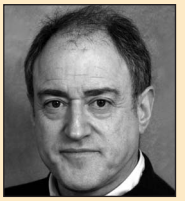
Acknowledgments

This work received support from the European Union Aspic project and the UK Economic and Social Research Council and Engineering and Physical Sciences Research Council (award L328253015). We're grateful to Richard Cooper, Tim Shallice, and Elizabeth Black for numerous discussions on the material in this article.

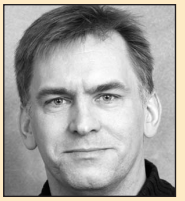
References

1. S. Das et al., "A Flexible Architecture for Autonomous Agents," *J. Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 4, 1997, pp. 407–440.
2. D.W. Glasspool, "The Integration and Control of Behavior: Insights from Neuroscience and AI," *Visions of Mind: Architectures for Cognition and Affect*, D.N. Davis, ed., Idea Group, 2005, pp. 208–234.
3. A.S. Rao and M.P. Georgeff, "BDI Agents: From Theory to Practice," *Proc. 1st Int'l Conf. Multi-Agent Systems* (Icmas 95), MIT Press, 1995, pp. 312–319.
4. J.E. Laird, A. Newell, and P.S. Rosenbloom, "SOAR: An Architecture for General Intelligence," *Artificial Intelligence*, vol. 33, no. 1, 1987, pp. 1–64.
5. J.R. Anderson et al., "An Integrated Theory of Mind," *Psychological Rev.*, vol. 111, no. 3, 2004, pp. 1036–1060.
6. J. Fox and S. Das, *Safe and Sound: Artificial Intelligence in Hazardous Applications*, AAAI and MIT Press, 2000
7. J. Huang, N. Jennings, and J. Fox, "An Agent-Based Approach to Health Care Management," *Int'l J. Applied Artificial Intelligence*, vol. 9, no. 4, 1995, pp. 410–420.
8. E. Black, "Using Agent Technology to Model Complex Medical Organizations," *AgentLink News*, Apr. 2003, pp. 7–8.
9. P. Tolchinsky, S. Modgil, and U. Cortes, "Argument Schemes and Critical Questions for Heterogeneous Agents to Argue over the Viability of a Human Organ," *Papers from AAAI Spring Symp.*, tech. report SS-06-01, AAAI Press, 2006; http://www.aaai.org/Library/Symposia/Spring/ss06-01.php.
10. J. Fox et al., "Towards a Canonical Framework for Designing Agents to Support Healthcare Organizations," to be published in *Proc. European Conf. AI (ECAI) Workshop Agents Applied in Healthcare*, 2006.
11. J.M. Spivey, *The Z Notation: A Reference Manual*, Prentice-Hall, 1989, p. 30.
12. S. Toulmin, *The Uses of Argument*, Cambridge Univ. Press, 1957.
13. M. Winikoff et al., "Declarative and Procedural Goals in Intelligent Agent Systems," *Proc. 8th Int'l Conf. Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 2002, pp. 470–481.
14. R.P. Cooper and J. Fox, "Cogent: A Visual Design Environment for Cognitive Modeling," *Behavior Research Methods, Instruments, and Computers*, vol. 30, no. 4, 1998, pp. 553–564.

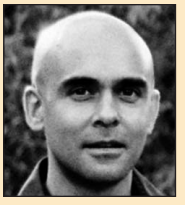
The Authors



John Fox is the director of the Advanced Computation Laboratory of Cancer Research UK, supervising a wide program of research in theoretical and applied AI. His research interests include clinical decision support systems and cognitive modeling. He received his PhD in psychology from the University of Cambridge. Contact him at the Advanced Computation Laboratory, Cancer Research UK, 44 Lincoln's Inn Fields, London WC2A 3PX, UK; john.fox@cancer.org.uk.



David Glasspool is a senior research fellow at the Advanced Computation Laboratory of Cancer Research UK. His research interests include executive control, routine behavior, and planning in the face of uncertainty, in both computational systems and human cognitive psychology. He received his PhD in computational modeling from the psychology department of University College London. Contact him at the Advanced Computation Laboratory, Cancer Research UK, 44 Lincoln's Inn Fields, London WC2A 3PX, UK; david.glasspool@cancer.org.uk.



Sanjay Modgil is a senior research fellow and project coordinator for the European Union Sixth Framework project, Argumentation Service Platform with Integrated Components. His research interests include formal models of argumentation and argumentation theory applied in a medical domain. He received his PhD in logic from Imperial College London. Contact him at the Advanced Computation Laboratory, Cancer Research UK, 44 Lincoln's Inn Fields, London WC2A 3PX, UK; sm@acl.icnet.uk.