SAND2012-9475C

Behaviors of Actors in a Resource-Exchange Model of Geopolitics

Curtis S. Cooper, Jacob A. Hobbs, Walter E. Beyeler, Michael Mitchell,
Z. Rowan Copley, and Matthew Antognoli

Sandia National Laboratories,
Albuquerque, NM, 87123 USA
{cscoope,jahobbs,webeyel,micmitc,mantogn}@sandia.gov,
zrcopley@smcm.edu (St. Mary's College of Maryland)

Abstract. We present initial findings of an ongoing effort to endow the key players in a nation-state model with behaviors. The model is based on resource exchange as the fundamental interaction between entities. In initial versions, model entities were severely limited in their ability to respond and adapt to changes in their environment. The ability to instill entities with behaviors of varying degrees of sophistication is essential for reliable policy analysis. To address this need, we have developed a hierarchical behavioral module, based on an extension of the proven ATLANTIS architecture, to provide flexible decision-making algorithms to agents. A Three-Layer Architecture for Navigating Through Intricate Situations (ATLANTIS) was originally conceived for autonomous robot navigation at NASA's JPL. It describes a multi-level approach to artificial intelligence. We demonstrate the suitability of our reification for guiding vastly different types of decisions in our simulations over a broad range of time scales.

Keywords: Complex systems, software engineering, agent-based simulations, artificial intelligence, robot planning, emergent behaviors, policy analysis.

1 Introduction

There is growing interest in the social sciences in numerically modeling the key processes and interactions important for determining change in geopolitical systems. A major long-term goal is to develop models to critically evaluate U. S. foreign policy in a complex arena of multi-national corporations, military and economic rivalries, and long-term changes in the balance of power between nations. To be useful in guiding policy decisions, models must capture the breadth of values and ideologies that guide nations of various sizes and political configurations.

The resource-exchange model [1] has been adapted to be capable of representing both historical and contemporary economic interactions among key players in the geopolitical

arena. The actors in the model can include national and sub-national entities (such as provinces) as well as corporations, whether sponsored by a single nation to advance its interests (e.g., the East India Company) or a semi-autonomous organization with international affiliations (e.g., Microsoft). We often refer to the model of [1] as the "nation-state model" in this configuration, which is well-suited for geopolitical scenarios. Although it does not yet include prediction of military conflicts and their outcomes, the model nevertheless captures essential features of real economic systems. For example, it has been used successfully to represent economic interactions in the current Pax Americana, in which global-scale military conflicts are inhibited.

The nation-state model structurally employs a systems' dynamics approach, in which differential equations representing the important rates of change in the system are integrated forward in time. The continuous timeline described by the rate equations is interspersed, however, with discrete resource-exchange events. These exchanges occur through markets and constitute the fundamental interaction between the entities in the model, which measure their success by a health attribute. Money is represented in this framework as simply a resource which can be exchanged for any other, whereas all other resources are exchanged in specific ratios according to their relative importance to different entities. Whether corporate, national, or sub-national, actors instantiate new processes and change parameters of processes already in operation in order to affect the flow of resources in and out of their boundaries. Processes represent what actors can do; what they *will* do is a separate question.

It is noteworthy that entities in the current implementation of [1] do not have the ability to make choices or analyze and adapt to their environments (e.g., to improve their performance). Their behavior is governed solely by the dynamics, with no awareness of the other entities in the simulation. The behaviors exhibited are therefore quite limited. In evaluating the strength of foreign policy decisions, naïve models carry the risk of underestimating the capabilities of adversaries and therefore evaluating U. S. policies over-optimistically. Actors in credible models must be capable of innovative approaches, such as trade regulations and the initiation of conflicts, to promote their own goals and undermine those of their competitors. Human history repeatedly demonstrates that nations initiate (or take actions that result in) conflicts in order to achieve political and economic ends. We are actively exploring theories of conflict and their effects to imbue agents with military capabilities.

It is the goal of this effort to implement agent behavior (with varying capability levels) for the resource-exchange model using ideas from the artificial intelligence community. Our approach has been to design and develop reusable Java language [2] modules, which themselves depend minimally on the structure of the resource-exchange model itself, in order to facilitate future application of the behavioral layer to other agent-based simulations. Architecturally, the software is a reification of the ATLANTIS architecture, proposed by [3], [4] as a solution for the complex problem of navigating NASA's ground rovers through rocky Martian terrain. We will discuss the advantages of their design for the applications anticipated for the nation-states model.

2 Simulating Agent Behaviors

2.1 Layers of Simulated Behavior

The ideas we present in this section come from internal discussions as well as perusal of the artificial intelligence literature, especially the excellent textbooks that have been written on the subject; see [5] and [6]. Defining intelligent decision-making processes by entities (herein also called agents or actors) in complex systems is critical to understanding the interactions between them, which can be both cooperative and competitive. The algorithms that accomplish this are expected to be broadly applicable. By not tying the behavioral layer to a particular complex-system model, we confer the ability for it to be used to inform decision-making for a wide range of agent-based systems. For example, flocking and steering behaviors can be encapsulated as common algorithms of certain animals. Once a particular behavior for an animal has been implemented, other animals exhibiting similar behaviors can be modeled using existing algorithms as a foundation.

For general utility in experimenting with multiple decision-making approaches (e.g., intelligent, scripted, random, etc.), it is desirable to have an interface separating the decision-making algorithms from the specific structures of the models to which they will be applied, to promote maximal reuse of code and ideas. It is also desirable to be able to provide certain entities with decision-making capabilities that are not cognitive, knowledge-based, or even particularly intelligent. For example, for evaluating the performance of a particular entity and its decisions, it may make sense to script the specific actions of its competitors. Similarly, in systems such as ant colonies, inter-agent cooperation is possible without cognitive decision-making by any one member. Individual ants behave reflexively to specific chemical stimuli they receive from others in the colony.

The first-order, rudimentary approach to modeling behavior, which has been used with some success in video games, is scripting. Scripted actors have a set strategy they follow. For example, a nation-state in our model could be told to always spend 10% of its GDP on military products, unless that action engenders a response from others. This approach, which is straight-forward to implement, can certainly instill nations and corporations with behavior. It is important to recognize, however, that scripted, predictable strategies, in which the behaviors of actors cannot adapt in time to new input and new situations, will almost certainly fail in competitive environments against adversaries that learn and adapt.

The next layer of sophistication for agent behavior is to instill actors with reflexes. For example, even animals we would not consider to be particularly intelligent have the ability to blink an eye to avoid being hit by a pebble. This rule-based approach to behavior is sufficient for making good decisions in some situations. For example, contemplation is not required (or a good idea) to avoid a car crash while driving. Braking and/or steering actions must be taken in a timely manner to preserve the driver's health.

The most sophisticated artificial intelligence systems are deliberative. That is, the decision-making process must evaluate multiple options and make choices in environments in which the state of the world is uncertain and the outcomes of actions are not entirely predictable. Although challenging to implement (and potentially intractable), deliberative decision-making may be needed in order to achieve the ultimate goals of this research: to begin evaluating computationally the performance of diverse policy choices in a multitude of hypothetical scenarios.

2.2 Components of Intelligent Agents

We assume here that the agents we will be dealing with in practice are neither omniscient nor omnipotent. They are limited in their knowledge (and their ability to acquire knowledge); and those agents that can perform actions (actors) have a discrete, limited set of capabilities to affect their surroundings. Some but not all actors will also have memories and the ability to learn from past experiences. For agents that learn and adapt to their surroundings, we do not necessarily expect repeatable output for a given sequence of inputs. Intelligent agents exhibit dynamic behaviors; they are capable of improving their performance at tasks by repetition.

We consider as a reasonable starting point a functional model for decision-making, which has the flexibility to encapsulate at least some common elements of the range of problems requiring agents to make decisions and perform actions. Note we conceive of decisions and actions (or plans of action) as separate concepts. The latter are clearly the output of any decision model, but the implementation of a plan is time-dependent and therefore susceptible to dynamical obstacles in the environment or disruption by the actions of other agents. In other words, things don't always go according to plan. Dynamic, intelligent agents should be capable of perceiving obstacles and rethinking plans as unforeseen circumstances arise.

Before discussing our proposed functional decomposition of decision-making problems, other temporal questions about decision-making problems should be mentioned. It seems likely that complex agents will perform multiple tasks simultaneously. How often do decisions about these tasks need to be reconsidered? What level of analysis or effort is appropriate for each decision type? How much time is available to make each decision before action is required for the well-being (even survival) of the agent? How is the timing of decisions related to the rate of sensory input and the agent's ongoing accumulation of knowledge of its environment? Answers to these questions are likely to be highly application-dependent. We note here only that how decisions are made, which it is our goal to address here, is distinct from when decisions are made. An application attempting to model the behaviors of intelligent agents must consider both problems in turn.

If the output of a decision-making model (or algorithm) is a plan of action, what is the input? To determine this, we must first separate those elements of the problem intrinsic to the agent and the elements extrinsic to the agent. The latter are the agent's environment, from which it receives input through its sensors and receptors in the form of *percepts*.

Environmental awareness is limited by the agent's sensors (which poll for new percepts) and receptors (which receive signals from the environment), and the time scales for updating the agent about the environment depend on how they function (e.g., ears and eyes, which allow animals to gather information at different frequencies).

The agent must have a self-model of its sensors, actuators, and world view, which specify the inputs to various decision-making problems. Note agents may use different decision models for different types of decisions. For example, human eyes blink reflexively to prevent damage to them, even though humans also make knowledge-based decisions about other topics, which require intelligence. For agents that acquire and accumulate knowledge, decisions are not always made in reaction to percepts directly. Rather, sensory information is acquired, filtered, processed, and stored in the form of memories. Cognitive decisions are made based on knowledge–some sort of world model–comprising an agent's fragmented memories about its environment and its experiences with the outcomes of past decisions. Certain percepts to be sure (such as an imminent car crash) will trigger immediate decisions at specific times. Intelligent agents, however, will also make decisions at various (in principle unpredictable) times, depending on the importance of the topic, the agent's self-confidence about its current plans, and the availability of new, relevant information.

2.3 Decision Problems and Thinking Agents

We hypothesize in this discussion that many important decision-making problems can be embodied in a *decision function*. The decision function can be represented abstractly with a decision model, which evaluates different actions against whatever (presumably domain-specific) criteria are appropriate for the problem at hand. From an object-oriented architecture standpoint, the decision model is likely to be an aspect of the agent attached using some sort of strategy pattern [7]. This design makes it possible for new ways of making decisions (for different types of problems) to be dynamically plugged into the agent. We elaborate here on the rationale for the simplified form of the decision function we have adopted in these interfaces.

The decision function clearly must take in the sequence of percepts from the agent's sensors acquired since the last time the agent processed sensory information (e.g., by acting on it or incorporating the new information into its world model). What other inputs are required to fully specify a decision problem? In addition to the environmental information (what the agent knows or can detect), we also will need a description of the agent's actuators; i.e., those things that the agent can do. A plan of action, which is the output of a decision function, will naturally consist of a sequence of actions within the set of actions possible (e.g., according to the agent's physics or other manifestation). The set of allowed actions limits the scope of each decision problem to a finite set of possible choices, although the correct plan for the agent might in some cases consist of an infinitely repeating sequence of actions.

For self-aware, thinking agents, we start with a simplistic model of human psychology rooted in the assumption of rational behavior: the so-called rational-actor hypothesis.

Rational-actor models assume intelligent agents always pursue their own interests, or what they believe to be in their own interests based on past experience, limited only by their imperfect ability to predict future outcomes of their actions. That is, they are goal-oriented and seek effective solutions to the problems they encounter that promote their goals. The filtering of information by sensory perception mechanisms in the agent lead to skewed information or incomplete knowledge about the system.

The rational-actor hypothesis affords great breadth of freedom to entities. The notion that judgments are subjective is captured by differences in values between entities; i.e., those concepts the entities view as priorities. The value sets can vary greatly between entities and are in principle time-dependent, varying with a given entity's perception of its surroundings. In rational decision-making, all options for an entity to pursue its interests are on the table. We do not assume motivations beyond the initial conditions and axes for evaluating decisions, which enumerate the (hypothetically orthogonal) set of ideas entities would value to varying degrees.

With a suitable decision function, artificial intelligence becomes a well-posed problem. Agents must combine their knowledge to solve problems in their task environments, with the potential to improve their performance with experience. In practice, the decision-making algorithm can consider multiple choices for actions (or plans of actions) against various goodness criteria. The problem is then optimizing the agent's path through the decision tree, considering tradeoffs and likely responses from allies and adversaries.

3 A Three-Layered Architecture

Many approaches have been tried since the inception of artificial intelligence to address the complexities involved in programming agents to solve problems (for example, see the discussion in [8]). In perusing various proposed solutions in the literature, we came to the conclusion that multi-layered approach would provide the flexibility we sought for the entities in the nation-state model.

TODO: finish the text for this section that describes the ATLANTIS architecture in more detail and the essential features shown in the figures.

4 Application of ATLANTIS

The solution offered by the ATLANTIS architecture offers numerous advantages over many others we investigated (e.g., a black-board architecture (briefly described in [5], p. 369-70). A full Java implementation of the components (even ignoring domain-specific objects and/or algorithms) is likely to involve a lot of code. But the architecture affords a sufficiently fine-grained division of labor amongst the components (without sacrificing flexibility) to allow the system to be built from the bottom up. Bottom-up construction is highly desirable because the individual pieces of the software are reasonably simple and testable.

The ATLANTIS architecture stays close to the objects in the problem domain itself and is therefore more intuitive than many other architectural descriptions for intelligent agents. It lends itself to development of a Java version without having detailed knowledge of the original version (or access to its source code, which was written in LISP by Erann Gat while at Virginia Tech and NASA's JPL; see [3, 4]).

The architecture also strongly decouples the parts of the problem that are challenging (deliberation in order to solve problems) from aspects that are more straightforward (agent control through sensors and actuators). In software-engineering terms, ATLANTIS describes objects with weak coupling but strong cohesion, which is a key principle of building maintainable software (as discussed at length in [7]). This decoupling of deliberation and control extends not just to code complexity but computational resources, since the deliberative layer runs asynchronously from the rest of the system and is allowed to consume an arbitrary amount of memory and processor time.

We furthermore consider it a major advantage of ATLANTIS that dealing with error conditions and reporting unknown situations is well-specified at the architecture level. It is the control layer's responsibility to detect the end state of the actions it attempts (through actuators) and report this information to the sequencing layer. Agents in ATLANTIS are failure cognizant. This implies that in a fully functional system, there is natural cohesion between the information-gather objects (sensors and receptors) and the effector objects (the actuators).

It is worth noting that the intermediate sequencing layer is potentially complex, depending on the application. Indeed, part of the design is to hide complexity in the sequencing layer. However, for a first pass-implementation, its operation can be approximated with a thread-safe priority queue (thread-safe because the deliberative layer runs asynchronously), in which the control layer has the option to interrupt in-progress actions at any time. We admit, however, that the sequencing layer is the main potential disadvantage to three-layered architectures. For complicated agents, it is conceivable that a robust implementation of the sequencing layer will prove elusive.

Nevertheless, a solution based on ATLANTIS has been deployed to solve a sophisticated robotics problem: how to autonomously steer a robot on the surface of another planet. Hence, the basic ideas are thought to be sound and compare favorably to alternative approaches to intelligent agent architectures (see [7]). Due to the generality of ATLANTIS and its dissection of the key interacting objects into manageable pieces, we expect the architecture to have broad applicability to complex systems problems, even though many of the anticipated applications are quite different from the robotics problem domain in which it was conceived.

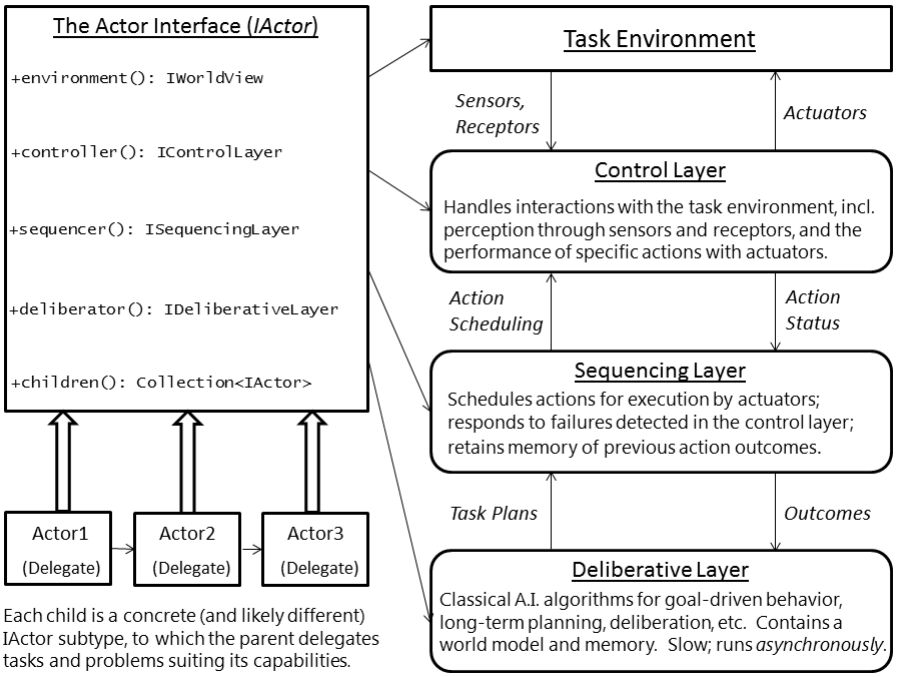

Fig. 1. As shown in this high-level overview of our Java reification of the ATLANTIS architecture [3], actors aggregate three interacting layers to make decisions accomplish tasks. The control and deliberative layers send messages to and from the mediating sequencing layer to coordinate tasks. The control layer is close to the environment and operates on short time scales, whereas the deliberative layer, which runs asynchronously, solves complex problems and suggests action sequences for queuing by the sequencing layer. Note also the hierarchical structure of the actors themselves; they are naturally organized as a composite. Hence, specific tasks either too complicated or too low-level for the agent can be delegated to child actors (the delegates in the diagram) under the supervision of the parent.

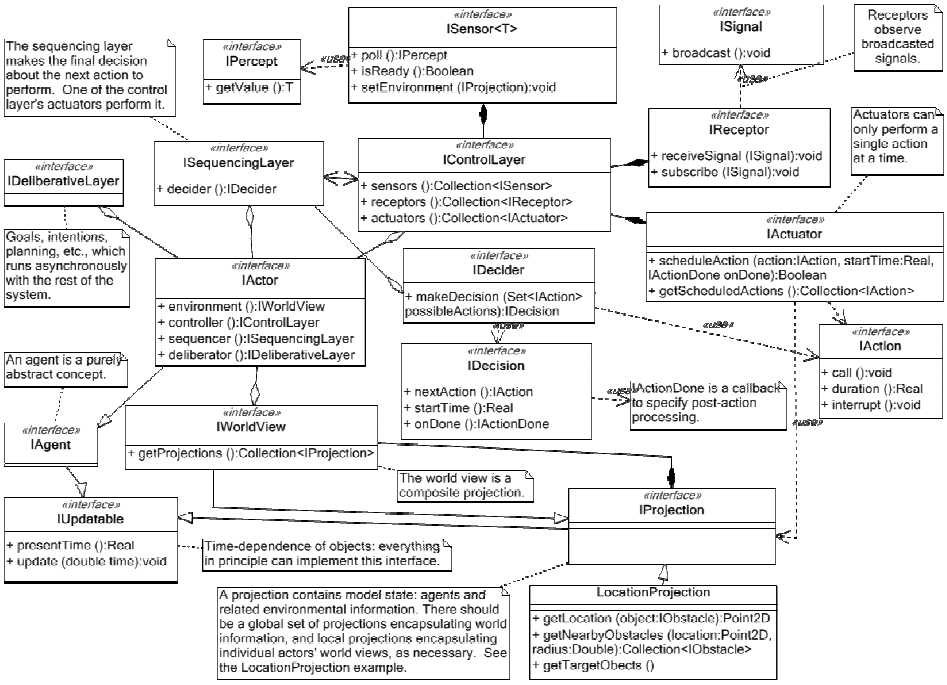

Fig. 2. In this UML diagram of the key objects involved in the ATLANTIS architecture for decision-making, the task environment is represented with composite projections (IProjection instances). We represent the notions of perceptions and beliefs with these projection objects, as the world view of an object can represent a distorted or incomplete description of the state of the system. All direct interaction of the agent with its environment occurs through the control layer.

References

1. Beyeler, W. E., et al.: "A General Model of Resource Production and Exchange in Systems of Interdependent Specialists", Sandia Report SAND2011-8887, (2011)
2. Schildt, H.: Java: The Complete Reference, Eighth Edition, McGraw-Hill, New York (2011)
3. Gat, E.: Reliable Goal-Directed Reactive Control for Real-World Autonomous Mobile Robots, Ph. D. thesis, Virginia Tech, Blacksburg, Virginia (1991)
4. Gat, E.: Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. (Proc. of the 10th AAAI Conf.). pp. 809–815 (1992)
5. Jones, T. M.: Artificial Intelligence: A Systems Approach. Jones and Bartlett (2009)

6. Russell, S.J., and Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd Ed., Pearson Education (2010)
7. Holub, A.: Holub on Patterns: Learning Design Patterns by Looking at Code, APress (2004)
8. Gat, E.: On Three-Layer Architectures, A.I. and Mobile Robots, AIII Press (1998)