IsisWorld: An Open Source Commonsense Simulator for AI Researchers

Dustin Smith and **Bo Morgan**
{dustin,neptune}@media.mit.edu
MIT Media Lab
20 Ames St
Cambridge, MA 02139 USA

Abstract

A metareasoning problem involves three parts: 1) a set of concrete problem domains; 2) reasoners to reason about the problems; and, 3) metareasoners to reason about the reasoners. We believe that the metareasoning community would benefit from agreeing on the first two problems. To support this kind of collaboration, we offer an open source 3D simulator containing everyday, commonsense problems that take place in kitchens. This paper presents several arguments for using a simulator to solve commonsense problems. The paper concludes by describing future work in simulator-based unified generative benchmarks for AI.

Introduction

Metareasoning is when a reasoner reasons about a reasoning system and is an important operation for solving problems with limited information and computational resources (Russell 1997; Cox and Raja 2008). Metareasoning is also viewed as a useful, if not essential, property of systems that are able to solve a wide range of problems. A metareasoning system involves three components:

1. a set of concrete problem domains, and
2. a reasoner, or ensemble of reasoners, that reasons to solve problems, and
3. a metareasoner, or ensemble of metareasoners, that reasons about the reasoner.

Traditionally the reasoner and metareasoner are layered, although there is nothing preventing a meta-circular arrangement (Schmill et al. 2008) nor the addition of other metareasoning layers (Minsky 2007; Singh 2005). Reasoners can operate in serial or parallel.

With the explicit goal of building intelligent agents, we consider the reasoner to be a planning system that works to select its next action (Russell and Wefald 1991). Although there are many ways to formulate a planning problem, in general, problems of planning can be described as "designing controllers that can map sequences of observations into actions so that certain goals are achieved (Bonet and Geffner 2001)." Using this planning terminology, metacognition has many uses: deciding which problems to solve, reorganizing knowledge of actions or the world state, reasoning about learning strategies, adding or removing actions, recognizing a missing skill, adding a goal of acquiring a skill, prioritizing goals, and reasoning about schemes of selecting methods to solve problems.

As there is no consensus about the particular planning system nor the problem domain, there is a major communicational challenge for metacognition researchers; though see attempts at constructing a unified view (Anderson and Oates 2007; Cox 2005). Seymour Papert said, "You cannot think about thinking, without thinking about thinking about something," suggesting that one should only study problem solving processes in tandem with concrete problems. It's simple to repurpose Papert's wisdom to metareasoning: you cannot think about metareasoning, without thinking about metareasoning about something.

We take Papert's advice as a harbinger of what is missing in the research of metacognition: a shared problem domain.

Using simulators to study AI problems

Although there are many fields that study human intelligence, the field of AI is characterized by its engineering goal of making computers behave intelligently. We argue that using a commonsense simulator can be useful for pursuing AI's goal because of the following reasons:

- **Understanding the problems better by engineering a mind**: We believe that the aggregate knowledge about human intelligence contains enough "pieces of the puzzle" that we should begin to build a integrated mind. Just as with a puzzle, we may need to start building it before we can detect how various fragments are connected and if any pieces are missing.
- **Seeing the whole mind at once**: While a linguist may focus on understanding how a sentence is constructed, an AI researcher must also explore *why* that sentence was generated, and *what* information it is trying to communicate (Schank and Birnbaum 1994). As the field stands, it

is difficult for a researcher in, say, the sub-field of planning to understand how his solutions constrain or otherwise influence another researcher's model in the sub-field of language processing or scene understanding.

- **Resource constraints are what make meta-reasoning important**: Consider the differences in how a *super-human AI* with gigabytes of memory could learn the multiplication table for all 3 digit numbers by simply memorizing a table containing $\approx 10^6$ assertions, versus a human, for whom memorization is more costly, who would represent equivalent knowledge by learning a *process* that could compute the solution digit by digit— albeit much more slowly than table-lookup. On top of storage required for the procedural description, the human then only needs to learn the outcomes for ways to multiply all 10×10 single digits—100 facts— or 50 rather, if the learner[1] exploits a symmetry: $A \times B = B \times A$.
- **Variable environment complexity**: Scene understanding and object recognition are unsolved problems, but simulators allow varying sensory and motor control granularities—to pretend these hard problems have been solved. If there is a "chicken and egg scenario", for instance when higher-level cognition depends on object recognition and object-recognition depends on higher-level cognition, then the solution may require alternating between development of both and meeting in the middle.
- **Thinking about meta-reasoning about something**: Again, using Papert's advice, having a particular problem domain facilitates clear thinking. Further, using everyday common tasks as problems allows researchers from different backgrounds to be able to easily describe their problems and solutions using a common language.
- **Visual representation for debugging**: AI ultimately is a large software engineering challenge: requiring lots of programming and debugging cycles. A 3D simulation can convey a lot of information about the character's behavior. This helps a programmer to test the performance of his software and communicate the results to other people.

Picking a reasoning problem

Our goal is to *build human-level intelligence* and we are not bothered by not carefully distinguishing "human intelligence" from "intelligences in general". Because we are interested in building AIs that are capable of understanding natural languages, which we take to be inexorably linked to human cognition, our approach demands at least the ability to simulate human linguistic and conceptual competencies. These, we believe, are acquired in service of solving and communicating descriptions about traditional commonsense problems. Furthermore, if the simulated world's problem domains are too far removed from "the kinds of tasks humans generally solve", then our natural languages may not give the agent the right words with which to label its experiences.

[1]See a related discussion at `http://web.media.mit.edu/~minsky/OLPC-1.html`.

Our problem domain for the simulator involves problems that are familiar to most cultures: domestic tasks that take place in a kitchen, such as cooking and eating. Consequently, there is no learning curve in between a researcher and his or her understanding of the problem domain.

In summary, we chose problem domains that we believe cut across the everyday "commonsense" reasoning problems that people experience. We believe this is appropriate for building intelligences that solve *human problems* and can describe them using *human languages*.

What is a commonsense reasoning problem?

Commonsense reasoning involves thinking about many different types of problems, which requires a large amount of approximate knowledge and knowledge about when it can be used. Also, a commonsense reasoner must decide when to stop pursuing one problem solving strategy in order to try another, or to reorganize goals in general.

Our simulator emphasizes multiple problem domains amenable to studying commonsense reasoning:

- **Body movement and dexterous manipulation**: Path planning in the simulator is necessary for moving an agent's body over and around physical obstacles in the environment; for example, to go through doors, around counters, and up and down stairs. When using arms and hands, tactile and visual feedback are required.
- **Problem solving in a relational domain**: Objects in the kitchen, such as the loaf of bread and knife are programed with ways that they can be "used" together: a primitive action that requires one of the objects to be in the agent's hand. For example, when the knife is used with the loaf of bread, a slice of bread appears in the agent's other hand. In this way, the agent can learn how to accomplish simple cooking tasks by experimenting with objects.
- **Multi-agent social communication**: Our primary agent, Ralph, does not exist in the simulation alone. He is accompanied by other agents, including his mother. The presence of a caregiver allows us to model the ubiquitous but complicated learning situation where a child learns in the context of a parent.
- **Dominion of objects**: When multiple agents are solving problems in the simulation, problems of dominion can arise. Who owns or controls an object is a complicated problem because some objects are destroyed in order to accomplish goals. For example, when Ralph's mother makes a piece of toast for herself and Ralph wants it to satisfy his hunger, she could choose to tell him to make his own piece of toast.
- **Visual reasoning**: Although the simulation is not meant to be a photo-realistic model of the physical world, there are many higher-level visual reasoning problems that could be studied in this simulator, including object permanence: when one object is seen and then passes behind another occluding object.

As we have elaborated, there are many different problem realms involved with even "simple" social and cooking tasks. In the field of AI, each of these different problems realms are studied independently, often by different people. We hope to see the tools of metacognition and reflective control used to organize and control a variety of problem solvers. Specifically, we see our physical simulation employing metacognition and reflective learning to achieve:

- **Transfer learning**: A skill or knowledge that is involved with one problem domain sometimes can be applied to other domains. For example, bodily knowledge is exploited in the service of reasoning about space: the front of a kitchen corresponds to a 'face', and the rear corresponds to its 'back'. English speakers extend spatial knowledge to time; consider some analogies: ***before*** *eating*, ***through*** *the week*, ***by*** *today*. Evidence of bodily word knowledge being used to describe increasingly abstract ideas such as space, time and mental states has been documented across languages (Deutscher 2005).
- **Knowing what knowledge is relevant**: Commonsense reasoning involves a lot of knowledge, so a reasoner must be selective about which knowledge it uses to solve a particular problem. Being able to select what knowledge is available to a planner is a metacognitive problem.
- **Reflective debugging of plans**: When planning under uncertainty, the credit assignment problem becomes necessary to solve from a metacognitive reasoner outside the planner.
- **Mental self-models and stories**: Because the world involves multiple agents that perceive and act in similar ways, they will accumulate shared experiences and knowledge. Problems that require multiple agents will benefit from using mental models of other agents. With a metareasoner, an agent can exploit his own self-models and episodes to reason about other agents' mental states.
- **Personality and self-reflection**: Knowledge about an agent's own abilities, mental and physical, are useful for coordinating multiple expert agents for solving hard problems and recognizing when new skills need to be acquired.

Introducing the simulator

Our simulator exists in a 3D world with basic physics of gravity and collision detection.

The simulator uses the open-source Panda3D game engine and is written in Python. The source code repository is public[2].

Perceptual and motor interfaces with the simulator

By default, the simulator is *paused*, and no physical simulation can take place. The simulator can be *unpaused* and stepped, advanced incrementally, or run continuously. The character in the simulator, "Ralph", can be controlled directly in the simulator using key commands or separately through a programmed agent (a client). The environment and agent are temporally decoupled. In this section, we review the agent-environment interface in terms of a **perceptual frame** and **primitive actions**.

Perceptual Frame The agent can *sense* the environment, which returns a perceptual frame containing:

Object Information All names of the *items* in the character's field of vision; each item's center's relative x and y coordinates on the frustum: the 2D plane representing the projected 3D region of the character's field of vision [3]; the item's area of the entire field of vision; the item's distance; and the item's global orientation.

Proprioceptive Information The character's global position (x, y, z) and orientation (h, p, r) in the environment as well as the relative positions and orientations of the character's head and limbs with respect to its body; and the names of the items within each of the character's hands (if any).

Linguistic Information A buffer representing all of the text entered by the user since the last time the sense command was executed.

While the granularity of actions and perceptions can be refined when necessary, we chose this level of abstraction to focus explicitly on solving everyday tasks and commonsense reasoning.

Primitive actions The character has a set of primitive actions that can be taken including: moving forward, backwards, left, and right; looking in four directions; turing in

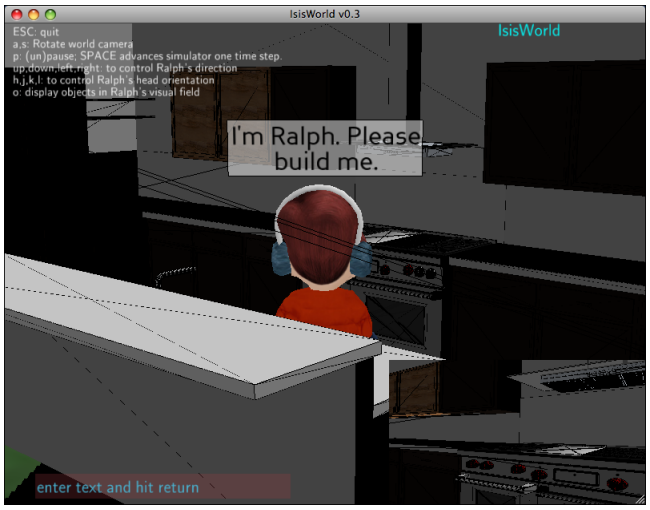

Figure 1: The simulator showing the agent's monocular field of vision in the lower right corner's picture-in-picture.

[2] It is hosted on GitHub.

[3] (0,0) denotes the center of the field of vision, where x and y are values in $(-1, 1)$

four directions; moving arms; picking up objects in either hand; rotating objects; dropping objects; and 'using' objects with one another.

Many actions can be performed simultaneously. Their compositional constraints can be encoded by a first-level reasoner (i.e., a planner) or discovered through experimenting and learning the effects of the actions, detecting when actions conflict, by a metareasoner.

Obtaining and running the simulator

Binaries and instructions for obtaining the simulator are available[4] along with a simple agent implementation. Binaries are currently supported for all major operating systems.

Going forward

AI has fragmented into many sub-fields and lacks a unified goal. The specific sub-fields of AI: classification in machine learning, object recognition, inductive logic programming, and so forth have all made progress in their own research communities. How can they be brought back together?

Generative canonical tasks within a simulator?

AI would benefit from some canonical and all encompassing problems that require syntheses between many different lines of work. Canonical tasks always run the risk of diverting resources away from other research, so designing them is a large responsibility. From past tasks, we know the task designers must assume that the researchers will "overfit" their solutions to meet the task's requirements—that the solutions will not generalize to other tasks, despite the original intentions behind creating the task (e.g., (Shannon 1950)). Further, if the task is too challenging or does not decompose into smaller parts (e.g., (Turing 1950)) researchers will have difficulty making progress.

How could we come up with a task that would prohibit shortcut solutions, which only solve the task but do nothing else? The task could be designed to make systems fail at many levels of problem solving (motor controls, perceptual limitations, goal conflicts, bad/incomplete information, abandoning some problem after spending too long on it, etc) to encourage flexibility. One possible way to do this is by continuously changing the task: e.g., keep adding more constraints at runtime. Another way is to make the problem not a particular task, but a *space of tasks*, where each particular task is drawn from a **task generator**.

Example of a cross-cutting task generation pipeline With the goal of an integrated simulator for studying planning, problem solving, and communication, we have come up with possible multi-step approach to generating tasks:

1. Select parameters and generate a simulated world.
2. Make the agent learn how to label things and states in the world, perhaps by automatically providing examples and counter examples.
3. Make the agent learn how to describe changes in states, induce changes in states, and learn which are desirable.
4. Make the agent communicate this knowledge to another agent, either human or simulated.
5. Evaluate this entire system on the quality of the communicated information, quantified by how well the recipient of the information was able to use it to accomplish a task.

Though we may be far from this stage, getting several different researchers to work together on a shared simulator environment is a start. Sharing tasks and evaluation metrics using the shared simulator are next steps.

[4] `http://web.media.mit.edu/~dustin/simulator_setup/`

References

[Anderson and Oates 2007] Anderson, M., and Oates, T. 2007. A review of recent research in metareasoning and metalearning. *AI Magazine* 28(1):12.

[Bonet and Geffner 2001] Bonet, B., and Geffner, H. 2001. Planning and control in artificial intelligence: A unifying perspective. *Applied Intelligence* 14(3):237–252.

[Cox and Raja 2008] Cox, M., and Raja, A. 2008. Metareasoning: A manifesto. *Proceedings of Metareasoning: Thinking about Thinking*.

[Cox 2005] Cox, M. T. 2005. Metacognition in computation: A selected research review. *Artificial Intelligence* 169(2):104–141.

[Deutscher 2005] Deutscher, G. 2005. *The unfolding of language: an evolutionary tour of mankind's greatest invention*. Macmillan.

[Minsky 2007] Minsky, M. 2007. *The emotion machine: Commonsense thinking, artificial intelligence, and the future of the human mind*. Simon and Schuster.

[Russell and Wefald 1991] Russell, S., and Wefald, E. 1991. Principles of metareasoning. *Artificial intelligence* 49(1-3):361–395.

[Russell 1997] Russell, S. 1997. Rationality and intelligence. *Artificial intelligence* 94(1-2):57–77.

[Schank and Birnbaum 1994] Schank, R., and Birnbaum, L. 1994. *Enhancing intelligence*. 72–106.

[Schmill et al. 2008] Schmill, M.; Oates, T.; Anderson, M.; Fults, S.; Josyula, D.; Perlis, D.; and Wilson, S. 2008. The Role of Metacognition in Robust AI Systems.

[Shannon 1950] Shannon, C. 1950. Programming a computer to play chess. *Philosophy Magazine* 41:256–275.

[Singh 2005] Singh, P. 2005. *EM-ONE: An Architecture for Reflective Commonsense Thinking*. Ph.D. Dissertation, Massachusetts Institute of Technology.

[Turing 1950] Turing, A. 1950. Computing machinery and intelligence. *Mind*.