A Testbed for Evaluation of Architectures for Physical Agents

Dongkyu Choi, Michael Morgan, Chunki Park, and Pat Langley
Computational Learning Laboratory
Center for the Study of Language and Information
Stanford University, Stanford, California 94305
{dongkyuc, mmmorgan, lovelive}@stanford.edu, langley@csli.stanford.edu

Abstract

While cognitive architectures provide excellent infrastructure for research stretching over various fields, the integrated nature consisting of multiple modules makes their evaluation extremely difficult. Due to the lack of analytical criteria, the cost of general demonstrations, and varying specifications among different architectures, deriving any general evaluation methods is a complicated task. In this paper, we propose a method for empirical evaluation, using a testbed within an in-city driving environment. With its familiar but challenging missions cast in a rich setting, the testbed provides a uniform and competitive environment for agents, for evaluation of cognitive architectures that embodied them.

Introduction

Cognitive architectures combine ideas from psychology, computer science, and other fields, and support research of general intelligence. They usually incorporate many different components that operate in combination to perform reasoning and problem solving, and learn new knowledge. For this reason, they provide excellent infrastructure for the research. However, it also makes evaluation of these architectures extremely difficult, partly due to the lack of analytical criteria and the cost of general demonstrations. In addition, different architectures have different specifications that support distinct capabilities, further complicating systematic evaluation.

Therefore, researchers in this field have mainly focused on empirical evaluation, based on performance and learning results of their architectures in several different test domains. Although this provided a way to estimate the capabilities of individual architectures, each group of researchers uses its own favorite domains, often developed in-house, and it has been very difficult to compare different architectures in terms of their capabilities and performance. There have been some efforts in the literature to resolve this issue, General Game Player (Love et al., 2006) and RoboCup (Kitano et al., 1997) being excellent examples. However, the former encodes knowledge in strict logical terms, making its application to physical domains extremely hard. The latter inherently focuses on physical aspects, but showed its shortcomings recently as participants tend to concentrate on details of particulars of the game.

In this paper, we propose a testbed for empirical evaluation that provides a uniform environment to test architectures for physical agents. The testbed also provides a competitive environment that supports side-by-side performance comparisons. Unlike the General Game Player which provides puzzle-like games, the testbed is designed specifically for architectures for physical agents. Also, thanks to the richness of the driving environment in a downtown setting, the testbed will allow agent developers to focus on high-level strategies that are interesting, once the basic driving behaviors are programmed properly. In the following sections, we will review and justify our motivations behind selecting the environment as our test domain, and provide an overview of the testbed. Next, we will provide detailed explanations on various evaluation criteria available. Then we will review related work and sketch our future plans before we conclude.

Motivations for an In-City Driving Domain

Many people make their living by driving for a variety of purposes. Some drive to commute, while others do to deliver packages, or to shuttle around people. Driving takes such an important part of our life, so we even saw a commercial for an automobile company that depicted how people's life begin with rides to hospitals right before their birth and end with rides to cemeteries for funerals. On the other hand, there are rules and common sense to follow in driving, for example, we only drive on paved or unpaved roads, not into buildings. Therefore, the domain is very familiar to people and very rich, but highly constrained.

Our motivation for a simulation of driving comes for this reason. For driving, there are so many different reasons and goals, and drivers encounter various situations involving many objects. In particular, in-city driving tasks require tremendous amounts of attention to even more factors, when compared to highway driving or even flying a small aircraft with much less traffic. Therefore, we chose a simulation of typical driving situations in downtown as our research domain. Here, we have various objects, including pedestrians, traffic signals, and other vehicles moving in different direc-

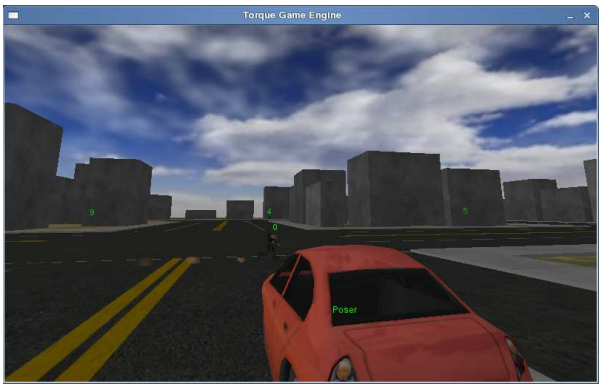

Figure 1: A screenshot of the in-city driving domain.

tions, to name a few. A driving agent needs to be attentive to these objects at all times while still maintaining safe and legal behavior. However, the domain does not require complicated manipulations, and necessary sensors are relatively simple.

This is why the in-city driving domain provides an interesting and rich environment to test different capabilities of agent architectures, while still being practical and feasible. Also, it is a very familiar environment to agent developers as they encounter similar situations everyday.

A Testbed for Evaluation of Architectures

Previously, we developed a two-dimensional version of an in-city driving domain, and used it as a testbed for our ICARUS agent architecture (Langley and Choi, 2006). Although the visualization was not up-to-date to satisfy the crowd used to three-dimensional games, it provided a rich environment with various objects an agent can interact with, including streets with lane lines, sidewalks, buildings with addresses, traffic signals and drone cars. The agent was given the task of driving a car in the city, working for goals like package delivery.

Although the domain provided an excellent testbed for our initial research purposes, we soon discovered its shortcomings, especially in its description of vehicle dynamics and the visualization of the environment. In this context, we started the design and implementation of our new in-city driving domain.

Implementation

To build this new testbed shown in Figure 1, we used *Torque Game Engine* (TGE), a three-dimensional game engine developed by GarageGames®. It provides convenient and well-built templates for three-dimensional games, as well as robust networking capabilities, a C++-like scripting language, a terrain generating tool, and other powerful tools for game developers. Also, the game engine allows us to develop new games in cross-platform computing environments including Linux, Mac OS X, and Windows.

The game engine provides a ready-made physics module for vehicles, that not only has realistic default parameters

Table 1: An example of object information an agent receives in the environment.

```
((SELF ME SPEED 0.0 WHEELANGLE 0 THROTTLE 0
  LIMIT 25 BRAKE 0 SEGMENT S1734 HITS 0)
 (STREET FIRST) (STREET SECOND) (STREET B)
 (INTERSECTION S1812 STREET B CROSS FIRST
  DIST 5.73842)
 (SEGMENT S1734 STREET B DIST 88.986)
 (SEGMENT S1767 STREET FIRST DIST 19.7416)
 (BUILDING B1659 ADDRESS 8 STREET FIRST
  C1ANGLE 82.8411 C1DIST 33.6707
  C2ANGLE 58.9024 C2DIST 40.6854)
 (BUILDING B1682 ADDRESS 1 STREET B
  C1ANGLE 45.4169 C1DIST 35.197
  C2ANGLE 32.5563 C2DIST 49.2716)
 (LANE-LINE WHITE1 COLOR WHITE
  DIST 11.65 ANGLE -84.8146 SEGMENT S1767)
 (LANE-LINE YELLOW2 COLOR YELLOW
  DIST 5.814 ANGLE 4.83383 SEGMENT S1734)
 (PACKAGE P1 ADDR 5 STREET A DELIVERED N)
 (PACKAGE P2 ADDR 1 STREET B DELIVERED N))
```

Table 2: Sample actions an agent can execute in the environment.

```
(*cruise)          :  NO OP
(*gas %pedal)      :  push the gas pedal
                      at the given amount
(*brake %pedal)    :  push the brake pedal
                      at the given amount
(*straighten)      :  straighten
                      the steering wheel
(*steer angle)     :  steer
                      at the given angle
```

but also allows user-defined settings. We used *TorqueScript*, a scripting language the game engine provides, for the interface between the new domain and our Lisp-based architecture, as well as several customized mission maps. The language supports sufficient complement of functions including mathematics, standard input/output, basic object manipulation, and other helper functions. Scripts written in this language does not need to be compiled with the main game engine, providing flexibility for many different applications.

Since our simulation is based on TGE, architectures can directly connect to the testbed in C++ language. However, the testbed also provides an interface for Lisp-based architectures. Through the interface, agent architectures can receive information on perceived objects in the environment, and execute actions in the world. Table 1 shows an example of such perception an agent can get from the world, and Table 2 gives some sample actions available in the environment.

Missions Supported

As people experience everyday, an agent in the domain can be given many different tasks. We can have it simply drive around without hitting any pedestrians crossing streets, or give additional tasks including,

- delivering packages to multiple addresses,
- picking up passengers and dropping them off at various destinations,
- chasing another car and issuing tickets like cops,
- driving an ambulance to hospitals, or
- joy-riding in a sports car as fast as possible.

These different missions require different parameters like mass, height, engine torque, and braking capability of the vehicle an agent will drive, and the testbed supports easy changes of these parameters through the interface provided.

All of these missions are intended to test capabilities of agents ranging from sensory-motor control to high-level decision making strategies. Also, some of them may show the effect of habitual or emotional behaviors in agents, just as people typically get affected by such factors while driving.

Evaluation Criteria

Anytime an agent is driving in the simulated city, the testbed stores key variables like the location, velocity, and heading of the vehicle at predefined intervals. Also, it records the numbers of mishaps like,

- running over pedestrians,
- collisions with other vehicles and buildings,
- speeding,
- driving on the wrong side of road, and
- traffic signal violations.

During post-processing, the basic driving behavior can be evaluated using these traces, providing fundamental measures of the quality of the agent's driving. In addition, the testbed supports one or more mission-specific evaluation criteria. Some of these are,

- delivery time and the number of delivery errors,
- time between pick up and drop off and the number of errors,
- time until the capture of other vehicles and the number of tickets issued,
- time between pick up and drop off of emergency patients
- average speed and the number of tickets received.

These criteria are designed to measure how successful the agent was during the missions in both continuous and discrete scales, providing additional evaluation measures for architectures.

Related Work

There have been efforts in the literature to provide testbeds for empirical evaluation of architectures. One of the most successful is RoboCup (Kitano et al., 1997), that started as an interesting testbed for artificial intelligence and robotics research. It has promoted research in these areas and held its own competitions for a decade. However, as the competition grow larger, the recent focus has been more towards winning techniques specific to the game, rather than fundamental research issues that were originally pursued.

The General Game Player (Love et al., 2006) is a more recent example, that has hosted competitions since 2005. With its description language in logic, the testbed provides complete descriptions of games to players, allowing them to focus on decision making processes. It provides a series of puzzle-like games that are interesting, but it is difficult to use when evaluating architectures in physical settings since it requires complete descriptions for physical simulations that are more complicated and often times even nondeterministic.

Future Work

Our work is still in its early stage, and we plan to continuously improve the in-city driving domain and cast it as a uniform testbed many developers can use for evaluation of their own cognitive architectures in a competitive environment. There can be many additional features to be implemented, but most important among them may be moving towards a multiagent setting. Currently the domain supports single agent missions only, restricting the scope of evaluation. With multiple agents simultaneously running in the world, possibly controlled by different architectures, more competitive missions can be introduced. Racing to flags, hide and seek other agents, and chasing each other are some examples.

In a more broad sense, we may integrate other physical domains to provide a variety of games and possibly promote research for knowledge transfer among them. This will allow us to evaluate learning and transfer capabilities of architectures better. Also, agent developers will benefit from multiple test domains available through a uniform infrastructure, by reducing the time and cost of test and evaluation across different domains.

Conclusions

Cognitive architectures provide an excellent infrastructure for research, but their integrated characteristics make their evaluation extremely complicated with analytical methods. For this reason, we proposed a testbed for empirical evaluation, cast in an in-city driving domain.

The domain is a familiar, but challenging environment for physical agents. Although it is rich with a lot of objects and many possible tasks, it is reasonably constrained for research purpose, with a small set of manipulators and relatively simple sensors. The domain allows various goal-oriented behaviors with many maintenance goals imposed. With multiple missions and corresponding evaluation criteria available, it

provides an excellent testbed for empirical evaluation of architectures. We hope we can package the domain in a more deliverable fashion, and make it available online in a near future.

Acknowledgements

Authors would like to thank GarageGames for allowing the free use of its products, including Torque Game Engine 1.4.2 for Linux, BTCP Car Pack, and Urban Pack 1.0.0 for our development.

References

Kitano, H. Kuniyoshi, Y. Noda, I. Asada, M. Matsubara, H. & Osawa, E. (1997). Robocup: A Challenge Problem for AI. *AI Magazine*, *18*, 73–85.

Langley, P., & Choi, D. (2006). Learning recursive control programs from problem solving. *Journal of Machine Learning Research*, *7*, 493–518.

Love, N. C., Hinrichs, T. L., & Genesereth, M. R. (2006). *General Game Playing: Game Description Language Specification* (Technical Report). LG-2006-01.