ALE Agent-Environment Protocol v1

Date: Tuesday, August 23, 2011

Author: Joel Veness

Overview

This document describes the external agent/environment communication protocol for agents interacting with ALE. By external we mean that both ALE and the agent are separate processes. The protocol is text based and draws inspiration from the Universal Chess Interface (UCI). While text based protocols are less efficient than binary protocols, they have the advantage of being human readable and making it much easier to write cross platform code.

# Protocol

Each agent directive is issued by sending a line of text to STDIN of the ALE process. Each environment response requires the agent to read a line of text from STDOUT of the ALE process. A typical way to setup communication between an agent and ALE is to have the agent read and write to STDIN/STDOUT and use a bidirectional pipe to marshal the data to and from the ALE process.

## Handshaking

ALE will initiate communication by first issuing a version command, which will be in the form

version <major>.<minor>

where <major> and <minor> are non-negative integers with the usual interpretation. For example, “version 1.2” would denote a more recent release than “0.2”. Similarly, “version 1.12” would denote a more recent release than “version 0.9”.

The agent then optionally sets one or more configuration options by issuing commands of the form:

option <name> <param1> <param2> …

where <name>, <param1>, <param2>, … are strings. The list of available options is described in a later section.

The agent finally signals that it is ready to begin the main communication loop by sending the command “ready” to STDOUT. If ALE can successfully process all of the requested options it will respond with “readyok”, otherwise it will respond with “error <message>” and exit. This concludes the handshaking process.

## Main Loop

After handshaking is complete, the agent and environment enter into a communication loop of the

Repeat:

ALE: <observation> <reward> | end

AGENT: <action>

where <action> is an integer between 0 and 17 inclusively, <reward> is an integer between -32000 and 32000 and <observation> is a string describing auxiliary information available to the agent that depends on the current observation mode.

# Configuration Options

### Observation Modes

The ALE platform supports a number of different ways to deliver observation data:

* raw – An array of color indices, with each index being a hexadecimal encoding of an integer between 0 and 255 inclusively. For example, FF0012 would represent the sequence 255, 0, and 18 of indices. This is the default observation mode.
* rleraw - A run-length encoded version of the RAW format. This can considerably reduce the amount of data needed to encode the observation. More specifically, each index is encoded within repeated 4 byte blocks. The first two bytes specify the index in hexadecimal, with the remaining two bytes specify the number of repeats (0 to 255) of the preceding index. If an index is repeated more than 255 times, it is encoded using multiple 4-byte blocks.
* ram - A snapshot of the current state of the Atari memory. This mode ensures that the observation is a Markov state, which is a prerequisite for many reinforcement learning algorithms.

Later versions of ALE will likely support more observation modes, incorporating features such as audio information or more sophisticated coding schemes.

# Example Sessions

## Example 1

ALE: version 1.0

AGENT: option obsmode raw

AGENT: option resolution 210 160

AGENT: ready

ALE: readyok

ALE: CCCCA443A212…6C212243 10

AGENT: 6

ALE: 3CCAA143A216…FC219241 0

AGENT: 12

…

ALE: end

AGENT: endok

## Example 2