cisc3665
game design
fall 2011
lecture # II.2
introduction to game AI and agents

topics:

- introduction to agents (continued)

references:

- notes on agents from *An Introduction to Multiagent Systems*, by Michael Wooldridge, Wiley (2002), chapter 1-2

formalisms for describing abstract agent architectures

- the set of possible environment states: $E = \{e, e', \ldots\}$
- the set of possible agent actions: $Ac = \{\alpha, \alpha', \ldots\}$
- a *run*, r, is a sequence of states and actions:

$$r : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \ldots \xrightarrow{\alpha_{u-1}} e_u$$

  and $\mathcal{R} = \{r_0, r_1, r_2, \ldots\}$, the set of all such sequences
- a run can end with an action: $\mathcal{R}^{Ac}$, or a run can end with a state, $\mathcal{R}^E$
- a *state transformer function* represents the behavior of an environment as it is effected by an action:

$$\tau : \mathcal{R}^{Ac} \to \wp(E)$$

  where $\wp$ is the power set of E, i.e., $E \times E$
- we make the assumptions that environments are *history-dependent* and *non-determimistic*
- if $\tau(r) = \emptyset$, then a run has ended ("game over")

- an *environment*, Env, is a tuple: $\langle E, e_0, \tau \rangle$, where $e_0 \in E$ is the initial state and τ is the transformer function
- an *agent*, Ag, is a function which maps runs to actions:

$$Ag : \mathcal{R}^E \to Ac$$

  and $\mathcal{AG}$ is the set of all agents Ag
- a *system* is a pair containing an agent, Ag, and an environment, Env
- associated with any system is a set of runs of Ag in Env:

$$\mathcal{R}(Ag, Env)$$

- we make the assumption that $\mathcal{R}(Ag, Env)$ contains only runs that have ended

- So, a sequence:

$$(e_0, \alpha_0, e_1, \alpha_1, e_2, \ldots)$$

  represents a run of agent Ag in environment $Env = \langle E, e_0, \tau \rangle$ if:

  1. e_0 is the initial state of Env
  2. $\alpha_0 = Ag(e_0)$
     and
  3. $e_u \in \tau((e_0, \alpha_0, \ldots, \alpha_{u-1}))$,
     where
     $u > 0$ and
     $\alpha_u = Ag((e_0, \alpha_0, \ldots, e_u))$

reactive agents

- a purely reactive agent decides what to do based only on the current state of its environment
- formally, an agent's *action selection function* is:

$$action : E \rightarrow Ac$$

is the choice of action taken by agent Ag

- for example, consider a *thermostat agent*:

$$action(e) = \begin{cases} \text{off} & \text{if } e = \text{temperature OK} \\ \text{on} & \text{otherwise} \end{cases}$$

agent perception: understanding the world around it

- the *see* function represents an agent's ability to perceive (sense) its environment maps *percepts* (i.e., sensor readings) to states:

$$see : E \rightarrow Per$$

where Per is a percept

- this allows an agent to determine the state of its environment based on its perceptions (i.e., its sensor inputs)
- we can say that an agent's choice of what to do (its action selection function) is a function that maps sets (or sequences) of perceptions to actions:

$$action : Per^* \rightarrow Ac$$

agent's internal state: understanding itself

- the agent's internal state, I, represents its perceptions, current and past (and can be used to record its history)
- so the action selection function for an agent that considers its history (unlike a reactive agent, which only "lives in the moment") is:

$$action : I \rightarrow Ac$$

- an agent updates its internal state using the *next* function:

$$next : I \times Per \rightarrow I$$

agent control loop

while(true) {
 $Ag.I \leftarrow i_0$ — $//$ Ag starts in internal state
 $Ag.see(e)$ — $//$ Ag observes its environment, e, and generates a percept
 $Ag.I \leftarrow Ag.next(i_0, Ag.see(e))$ — $//$ Ag updates its internal state
 $\alpha_0 \leftarrow Ag.action(Ag.I)$ — $//$ Ag decides what to do
 $Ag.do(\alpha_0)$ — $//$ Ag performs action α_0
}

utility

- when agents decide what to do, one thing they may consider is the *utility* of a state
- a utility function maps environment states to real numbers:

$$u : E \rightarrow \Re$$

- the utility of a run can be computed in various ways:
  - the *minimum* utility of all states in the run
  - the *maximum* utility of all states in the run
  - the *average* utility of all states in the run
  - the *sum* of all utilities of all states in the run
  - etc
- determining the utility of a run is a hard problem and is typically domain dependent

- instead of thinking about somehow combining the utilities of all the states in a run, think of a real number that can be assigned to the value of the run as a whole:

$$u : \mathcal{R} \rightarrow \Re$$

- and in this way, we can abstract away the detail of how to compute the utility for the run (which is hard and often is domain dependent)
- another hard problem is determining what numbers to assign for utilities of states or runs
- again, this is domain dependent
- we will not answer this question today, but we'll come back to this question later in the term

expected utility

- the probability that run r (a specific sequence of states and actions) occurs when agent Ag is placed in environment Env is represented as:

$$P(r \mid Ag, Env)$$

- the sum of the probabilities of all runs for an agent in an environment is:

$$\sum_{r \in \mathcal{R}(Ag,Env)} P(r \mid Ag, Env) = 1$$

- an agent might want to consider the *expected utility*, EU, of a given run:

$$EU(r) = u(r)P(r \mid Ag, Env)$$

- or we might want to assess the expected utility of an agent in a given environment:

$$EU(Ag, Env) = \sum_{r \in \mathcal{R}(Ag,Env)} u(r)P(r \mid Ag, Env)$$

example

- given environment $Env = \langle E, e_0, \tau \rangle$, defined as:
  $E = \{e_0, e_1, e_2, e_3, e_4, e_5\}$, $\tau(e_0 \xrightarrow{\alpha_0}) = \{e_1, e_2\}$, $\tau(e_0 \xrightarrow{\alpha_1}) = \{e_3, e_4\}$
- given two agents:
  $Ag_1(e_0) = \alpha_0$ and $Ag_2(e_0) = \alpha_1$
- given probabilities associated with the various runs:
  $P(e_0 \xrightarrow{\alpha_0} e_1 \mid Ag_1, Env) = 0.4$
  $P(e_0 \xrightarrow{\alpha_0} e_2 \mid Ag_1, Env) = 0.6$
  $P(e_0 \xrightarrow{\alpha_1} e_3 \mid Ag_2, Env) = 0.1$
  $P(e_0 \xrightarrow{\alpha_1} e_4 \mid Ag_2, Env) = 0.9$
- given the utility function:
  $u(e_0 \xrightarrow{\alpha_0} e_1) = 8$
  $u(e_0 \xrightarrow{\alpha_0} e_2) = 11$
  $u(e_0 \xrightarrow{\alpha_1} e_3) = 70$
  $u(e_0 \xrightarrow{\alpha_1} e_4) = 9$
- what are the expected utilities of the agents for this utility function?

optimal agents

- an optimal agent *maximizes* its expected utility:

$$Ag_{opt} = \arg\max_{Ag \in \mathcal{AG}} EU(Ag, Env)$$

- but it may not be practical (or possible) to compute this, e.g., if $Ag : \mathcal{R}^E \rightarrow Ac$ is really big
- so we consider a *bound* on the computation, m, and define: $\mathcal{AG}_m$ as the set of agents that can be computed on machine (computer) m
- then we define a *bounded optimal* agent Ag_{bopt} as:

$$Ag_{bopt} = \arg\max_{Ag \in \mathcal{AG}_m} EU(Ag, Env)$$

tasks

- a *predicate task specification* is a special case of assigning utilities to a run where:
  $u(r) = 1$ (meaning "true") $\Rightarrow$ agent *succeeds*
  $u(r) = 0$ (meaning "false") $\Rightarrow$ agent *fails*
  and we define the predicate task specification $\Psi : \mathcal{R} \rightarrow \{0, 1\}$
- a *task environment* is a pair $\langle Env, \Psi \rangle$
- a task environment specifies:
  - the properties of the agent's system
  - the criteria by which the agent's success or failure will be assessed
- the set of all runs of agent Ag in Env that satisfy Ψ is:

$$\mathcal{R}_\Psi(Ag, Env) = \{r \mid r \in \mathcal{R}(Ag, Env) and \Psi(r) = 1\}$$

- and we can say that Ag succeeds in Env if all its runs produce utility $= 1$, i.e.,:

$$\mathcal{R}_\Psi(Ag, Env) = \mathcal{R}(Ag, Env)$$

- we might be interested in predicting the probability that an agent will be successful in a particular environment:

$$P(\Psi \mid Ag, Env) = \sum_{r \in \mathcal{R}_\Psi(Ag, Env)} P(r \mid Ag, Env)$$

to do

- read ch 2 of Wooldridge book (handout)
- work on assignment for unit I (labI.2), which is due on SEPT 25
  (electronic submission instructions are forthcoming...)