

Seven Days in the Life of a Robotic Agent

W. Chong^a, M. Anderson^b, Y. Okamoto^{bc}, D. Perlis^{ab}

mailto: {yuan,mikeoda,yoshi,perlis}@cs.umd.edu

website: <http://www.cs.umd.edu/projects/active>

(a). Department of Computer Science, University of Maryland, College Park

(b). Institute for Advanced Computer Studies, University of Maryland, College Park

(c). Linguistics Department, University of Maryland, College Park

1. Self-improving agent.
2. An example scenario.
3. Computational Reflection.
4. Continual computation.
5. Active Logic.
6. Conclusion.

Self-improving Agent [1]

- If we can capture the concept of self-improvement in a general way, then it's conceivable that most of the known machine learning methods will follow “logically” .
- If we can make an agent to improve itself, then we don't need to improve it ourselves.

Self-improving Agent [1.1]

Needs (among other things):

- Model of itself, to reason about its own behavior.
- Performance model of itself to make sense of “improvement”. (Performance: throughput of tasks accomplished per unit of time.)
- Ability not only to reason *about* time, but also to understand that its reasoning and actions occur *in* time.

Example: Seven Days in the Life of an Office Robot [2]

- Job: delivering coffee, documents, etc., around an office building.
- Goal: making people happy (fulfilling as many requests as possible).

1st day

[2.1]

- The Robot was first given a tour of the building.
- Shown power outlets scattered around the building for recharging.

- Everything went well at first: delivered everything on target.
- Robot found itself unable to move — a low battery.
- The Robot knew that
 1. it needed power to operate;
 2. it could recharge itself to restore its battery;but it had never occurred to the Robot that, “it would need to reach an outlet before the power went too low for it to move!”
- Stuck for the day — failure (caffeine deprived computer scientists are not happy human beings)!

- Having learnt its lesson, the robot decided to find an outlet a few minutes before the battery got too low.
- Unfortunately, optimal path planning is an NP-complete problem.
- Power ran out when it found the optimal path.
- Stuck again! Surfing the web and learnt about “Deadline-Coupled Real-time Planning”.
- Planning takes time, and it couldn’t afford to find an optimal plan when its action is time critical.

- Decided to quickly pick the outlet in sight when its battery was low.
- Unfortunately, the outlet happened to be too far away, and robot ran out of power again before reaching it.
- There were only a few places it frequented; it could actually precompute the optimal paths from those places to the nearest outlets.

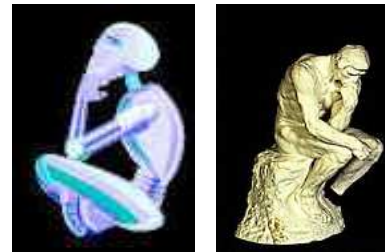
- Derived a theorem: “If the battery power level is above 97% of capacity when the Robot starts to move from any point in the building (and nothing bad happened along the way), it can reach an outlet before the power is exhausted.”
- Finally, avoid getting stuck; but still not responsive.
- Robot spent most of its time around the outlets recharging itself — since the Robot’s power level dropped 3% for every 10 minutes, the theorem above led it to conclude that it’d need to go to the outlet every 10 minutes.

- Knowledge base was populated with millions of theorems similar to the one it found the day before, but with the power level at 11%, 12%, ..., and so on.
- In fact, the theorem happens to be true when the power level is above 10% of capacity.
- Metarule: “a theorem suitably subsumed by another is less interesting; hence should be discarded.”
- With this more accurate information, the robot concluded that it could get away with recharging itself every 5 hours.

7th Day

[2.7]

That happened to be Sunday. Nobody was coming to the office. The Robot spent its day contemplating the meaning of life.



- Classical deductive reasoning, goal directed behavior.
- Abductive reasoning (diagnoses of failures).
- Explanation-based learning (derivation of recharging rules, compilation of navigation rules).
- Time-sensitivity (understanding that deliberations take time, people don't like waiting, etc).
- Reflection (revision of recharging rule, examining and reasoning about its power reading).

- A computational system is said to be reflective when it is itself part of its own domain, and in a causally connected way (Maes, 1988).
- More precisely, this means that
 1. the system has an internal representation of itself;
 2. the system can engage in both “normal” computation about the external domain and “reflective” computation about itself;
 3. reflection provides a principled mechanism for the system to modify itself in a profound way.

- Exploits the *idle time* of a computation system (Horvitz, 2001).
- Generalizes the definition of a *problem* to encompass the uncertain stream of challenges faced over time.
- Utility based approach presumes predictability.
- Perhaps we need a model closer to how mathematicians or theoreticians work.
- Theorems mathematicians derived can be seen as a form of precomputation.

We won't be able to show you an "axiom of self-improvement" yet; but we're working on the substrate to make it expressible.

- Capable of both forward and backward chaining;
- Can reason about time as well as in time;
- Can express higher order concepts such as contradiction, and deal with conflicting data.
- Need to better control its inference (perhaps through some reflective mechanism);
- Need to acquire "taste" to filter out unimportant lemmas.

Some Existing Accomplishments in Active Logic [5.1]

- Deadline sensitive planning and acting (Nirkhe et al, 1997);
- Detection and correction of misunderstanding in dialogue (Traum et al, 2002);
- Reasoning in the presence of contradictory data (Purang, PhD dissertation, 2001);

Some Formal Inference Rules in Active Logic [5.2.1]

Clock update:

$$i : \text{Now}(i)$$
$$i + 1 : \text{Now}(i + 1)$$

Contradiction detection:

$$i : P, \neg P$$
$$i + 1 : \text{contra}(\ulcorner P \urcorner, \ulcorner \neg P \urcorner, i)$$
$$\text{distrust}(\ulcorner P \urcorner, i)$$
$$\text{distrust}(\ulcorner \neg P \urcorner, i)$$
$$\text{distrust}(\ulcorner Q \urcorner, i)$$

where Q is a consequence of P or $\neg P$

Some Formal Inference Rules in Active Logic [5.2.2]

Extended Modus Ponens with Inheritance:

$$i : A, A \rightarrow B, B \rightarrow C$$

$$i + 1 : A, A \rightarrow B, B \rightarrow C, B$$

$$i + 2 : A, A \rightarrow B, B \rightarrow C, B, C$$

- We have described the (self-)development of a self-improving robotic agent, which hopefully can provide us some guidance in designing such an agent.
- We highlighted two key issues: computational reflection and continual computation, which we think especially worthy of exploring.
- It seems that we'll need a highly expressive symbolic system as a substrate to build such an agent.

Questions

[7]

- ?