The 5-and-10 problem and the tiling agents formalism (Brief technical note)

Benja Fallenstein

This document is part of a collection of quick writeups of results from the December 2013 MIRI research workshop, written during or directly after the workshop. It describes work done by the author, building on previous work by Vladimir Slepnev, Eliezer Yudkowsky, and others.

Consider the decision criterion from the tiling agents paper (Yudkowsky & Herreshoff, 2013): An agent Ag^i using proof system S_i only takes an action $a \in \mathrm{Acts}^i$, where Acts^i is the set of actions the agent is able to perform, if it has found a proof in S_i that $\overline{a} \to \mathcal{G}$, where $\overline{a}$ means that the agent takes action a, and $\mathcal{G}$ means that some goal is achieved; to summarize,

$$\forall a \in \mathrm{Acts}^i. \quad \overline{a} \quad \Longrightarrow \quad S_i \;\vdash\; \overline{a} \to \mathcal{G}.$$

Formally, Ag^i can be specified as a computer program which will return the $a \in \mathrm{Acts}^i$ the agent chooses to perform; then $\overline{a}$ is the formula in the language of arithmetic stating that this computer program halts and returns the value a. (To pick an extremely simple example, we could have $\mathrm{Acts}^i = \{0, 1\}$ and $\mathcal{G} = \overline{1}$; i.e., the agent can perform two actions 0 and 1, and its goal is achieved if and only if it chooses to perform action 1.) Note that the agent needs to know which theorem it is supposed to prove, and the target theorem makes use of the proposition $\overline{a}$ which refers to the agent's source code; thus, the agent's source code needs to refer to the agent's source code, but this is possible by quining.

To have a computer program P which searches for a proof of a theorem of the form "If P returns 1, then A" (for some proposition A) is very close to Vladimir Slepnev's logic-based models of updateless decision theory (see Slepnev's "What a reduction of "could" could look like").

However, there is an important difference, which makes Slepnev's system vulnerable to what is called the "5-and-10 problem", whereas Yudkowsky and Herreshoff's system is not. Essentially, this is because the latter system only considers what happens *if the action is taken that the agent actually ends up taking*, whereas Slepnev's system also considers what *would* have happened if the agent had taken a different action.

Using symbols similar to the tiling agents paper, Slepnev's system tries to prove a theorem of the form $\overline{a} \to U = n_a$, where each n_a is a numeral, for every $a \in \mathrm{Acts}$, and then takes the action a with the highest n_a (U is a term specifying the agent's utility function, such that $U = 7$ means that the agent achieves utility 7). For example, we could have $\mathrm{Acts}^i = \{5, 10\}$ and (i) $S_i \vdash \overline{5} \to U = 5$

and (ii) $S_i \vdash \overline{10} \to U = 10$. Then our intention is that our agent will take action 10 after having proven theorems (i) and (ii). However, if our agent in fact takes action $a \in \text{Acts}^i$, then $S_i \vdash \overline{a}$ and hence $S_i \vdash \neg\overline{b}$ for every $b \neq a$; thus, it's not obviously inconsistent to have our agent prove $S_i \vdash \overline{5} \to U = 5$ and $S_i \vdash \overline{10} \to U = 1$, and therefore take action 5, since this makes $\overline{10} \to U = 1$ vacuously true. Slepnev has constructed a concrete example of an agent, using a sound proof system S_i, that uses this reasoning to justify taking the bad action 5 (see his "An example of self-fulfilling spurious proofs in UDT"). Yudkowsky and Herreshoff's system does not have this problem because in order to take action a, it only has to prove that $\overline{a} \to \mathcal{G}$, and $\overline{a}$ is true for the a that the agent does in fact take.

(Vladimir Slepnev's solutions: Playing chicken and "A model of UDT without proof limits")

In short, the problem arises in Slepnev's system because it is a utility *maximizer*, which needs to compare the utilities of different actions. A utility *satisficer*, which merely shows that the utility achieved is larger than some fixed number, would be of the same structure as Yudkowsky and Herreshoff's system (e.g., setting $\mathcal{G} \equiv U \geq 7$), and therefore avoid the 5-and-10 problem. Utility maximization would be difficult in practice, since realistic agents usually won't be able to *find* the optimal action in the first place, much less prove that this action is in fact optimal, so it might seem as if the problem goes away as soon as we consider more realistic agents.

However, a similar problem arises when we consider utility "*meliorizers*", by which we mean an agent which has a built-in "fallback" option $a_0 \in \text{Acts}^i$, and which will take acton a_0 unless it can find, within a certain time limit, a proof that some particular other action a yields higher utility than a_0. The obvious way to implement this in Yudkowsky and Herreshoff's formalism is very similar to Slepnev's system, and suffers from the 5-and-10 problem for the same reason: We require that the agent choose an action $a \in \text{Acts}^i$ and find proofs $S_i \vdash \overline{a} \to U = m$ and $S_i \vdash \overline{a_0} \to U = n$, for particular numerals m and n such that $m > n$. Setting $a_0 = 10$ and $a = 5$, this is vulnerable in the same way as Slepnev's system, allowing an agent to prove $S_i \vdash \overline{a_0} \to U = 1$ and hence to take action a, thus making $\overline{a_0}$ false and the implication $\overline{a_0} \to U = 1$ (vacuously) true. Slepnev's construction of a concrete agent producing such a self-fulfilling "spurious" proof is easily adapted to this setting.