FALLENSTEIN'S MONSTER (BRIEF TECHNICAL NOTE)

NATE SOARES

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 mainly by Benja Fallenstein and Nate Soares.

1. Problem statement

Two known solutions to tiling agents (in the face of Löbian obstacles) have significant disadvantages. Marcello's Waterfall can only be realized in nostandard models, while Parametric Polymorphism is constrained to deploy different amounts of "proof strength" at different "successor depths". These concerns are explained in detail below.

We present a third solution to the problem, created by grafting Marcello's Waterfall and Parametric Polymorphism together. Our solution also works with a variant of Marcello's Waterfall which is sound on the standard natural numbers, described in a parallel note.

2. Marcello's Waterfall

Briefly, Marcello's waterfall works as follows:

Define $\psi(n)$ to be "n does not encode a proof of contradiction in ZF".

Statement 1. $\psi(n) := \neg Proof_{ZF}(n, \ulcorner \bot \urcorner)$

Then define a series of theories that each trust the following theories for as long as they can trust $\psi(n)$.

Statement 2. $T_i := PA \cup_\phi \{\psi(n) \rightarrow \Box_{T_{i+n^+}} \ulcorner \phi \urcorner \rightarrow \phi\}$

Marcello's waterfall is the first theory in this sequence, i.e.

Statement 3. $MW := T_0$

Assuming that ZF is consistent, $\psi(n)$ will never be false. Thus, we have a "waterfall" of theories that each trust the next theory. MW can accept proofs in T_1, T_2, T_3, and so on. Furthermore, MW is consistent. (The proof of consistency is subtle, but essentially follows from the fact that MW is far too weak to prove ZF inconsistent.)

We can use MW to build tiling agents (each of which accept proofs proven by their successor agents). The condition for action in such agents looks like this:

Statement 4. $\bar{a} \implies MW \vdash \bar{a} \rightarrow G$

Here, a denotes an action, $\bar{a}$ denotes the execution of that action, and G denotes some goal-state being achieved.

Consider an agent A_0 reasoning in MW and considering the construction of an agent A_1 running T_1. If A_0 can verify that A_1 only acts when it has proved that its actions achieve the Goal, then because T_0 can trust proofs in T_1, A_0 can deduce that the construction of A_1 will actually lead to achieving the goal. (See the Tiling Agents paper for formal details.)

Although MW is a moral victory over the Löbstacle, it comes with a few drawbacks. Among them is the fact that $MW \vdash \exists n : \neg\psi(n)$. MW has only nonstandard models (indeed, models where ZF is inconsistent). This suggests that agents running MW may be unfit for reality, wherein we believe ZF to be consistent.

Furthermore, MW succumbs to the procrastination paradox, which will be described later.

3. Parametric Polymorphism

Briefly, parametric polymorphism is an extension of PA (augmented with a new symbol κ) defined as follows:

Statement 5. $PP := PA \cup_\phi \{\kappa > 0 \rightarrow \Box_{PP} \ulcorner \phi \urcorner \rightarrow \phi[\kappa \setminus \kappa^-]\}$

In other words, whenever κ is positive PP is willing to trust PP-proofs with one caveat: If shown a PP-proof of ϕ, PP can deduce the statement $\phi[\kappa \setminus \kappa^-]$ (ϕ with all instances of κ replaced by $\kappa - 1$).

We can construct tiling agents using PP in a world with discrete timesteps. Such agents trust proofs made by their successors, with the caveat that when their successor proves an action safe for κ steps the parent concludes that the action is actually safe for κ^- steps.

In order to express this, we use a predicate $G(t)$ expressing that the goal has been achieved at timestep t. The condition for action then looks soething like this:

Statement 6. $\bar{a} \implies PP \vdash \bar{a} \rightarrow \forall t \leq \kappa : G(t)$

We require that subagents constructed by PP agents don't start running until the following timestep (and that PP agents know this). Then a PP agent is licensed to construct any other PP agent that proves its actions safe for κ steps: from $\Box_{PP}\ulcorner Safe(\kappa)\urcorner$ a PP agent can deduce $Safe(\kappa - 1)$, and can conclude that the action "build an agent which is safe for $\kappa - 1$ steps and run it in the next step" is safe κ steps. (For formal details, see the Tiling Agents paper.)

Parametric Polymorphism has standard models, and indeed $\{\mathcal{N}, \kappa = n\}$ is a model of PP for all n. However, PP has its own problems.

Parametric Polymorphism gets "stronger" when reasoning about other PP systems. Here, we only give an informal discussion.

PP itself does not trust the soundness of PA, for PP has no access to the term κ. (PP may never deduce $\kappa > 0$.) However, when a PP agent is reasoning about the safety of its subagents, its reasoning looks something like "Either κ is zero and this subagent will never run, or κ is positive and I can prove ϕ about the subagent".

The problem here is that *conditioned upon κ being positive*, PP can use $\Box_{PA}\ulcorner\phi\urcorner \rightarrow \phi$ for any ϕ not containing a κ. In other words, when a PP agent is reasoning about the actions of its successor agents, it can use $PA + 1$ (PA plus a soundness schema for PA).

However, when PP agents are reasoning about their own actions, they cannot condition upon κ being positive, and they can only reason in PA.

As you may expect, when reasoning about sub-subagents, PP agents can use $PA + 2$. In general, when reasoning about subagents n levels deep, PP agents can reason in $PA + n$. This is strange: agents running on PP can essentially deploy $PA + \omega$, but they can never deploy their full strength. The amount of "proof strength" that a PP agent can deploy depends upon what level of subagent it is reasoning about.

This is uncomfortable, to say the least. If we are comfortable with PP deploying PA plus many soundness schemas it should be able to use the strength of PA plus many soundness schemas when reasoning about its own actions in the immediate timestep.

4. The Procrastination Paradox

The procrastination paradox occurs when an agent has a goal that it delagates to its subagents ad infinum, such that the goal never gets achieved. The paradox is that certain tiling agent architectures with seemingly-sane conditions of action prove that this delegation is "safe".

For example, consider a goal requiring that a button is pressed at some point in time (it does not matter when). Imagine an agent reasoning *I can either press the button myself, or build a trustworthy child with the same goal.* It's not hard to see how such reasoning can lead to an infinite chain of descendents, none of which actually press the button.

The procrastination paradox is explained in detail elsewhere, for now it suffices to know that Marcello's Waterfall succumbs to the procrastination paradox. Parametric Polymorphism can be made to avoid the paradox, but some new machinery will be required.

5. Subdividing the Goal

Parametric Polymorphism can avoid the procrastination paradox with some clevel manipulation of the goal predicate. Naively, we might try a goal predicate like

Example 1. $G(t) = \exists u.\overline{b(u)}$

where $\overline{b(u)}$ denotes that the button b is pressed at timestep u. Intuitively, this goal requires that at every timestep the agent can prove that there is some timestep when the button becomes pressed.

This proves too restrictive: the initial PP agent (unaware of whether κ is positive) must press the button immediately, and so this goal is no better than $G(t) = \overline{b(0)}$. This is a rather unsatisfactory solution to the procrastination paradox, to say the least.

We can relax the goal somewhat to allow PP agents to construct successors who must eventually achieve the goal by restructuring the goal predicate as follows:

Example 2. $G(t) = t > 7 \rightarrow \exists u.\overline{b(u)}$

This allows seven timesteps before agents must prove that there is a time at which the button is pressed. Which such a goal scheme we

can allow agents to build up to seven successors, the last of which must actually make sure that the button becomes pressed. The human programers, in dividing up the goal, must put a finite limit on the allowed successor depth.

We would like to be able to express goals of the form "successor chains may be arbitrarily large, but must be finite".

We can extend Parametric Polymorphism to allow it to express such goals. Intuitively, we will do the following: instead of baking in a fixed maximum successor depth, we will index goals using proof theoretic ordinals and require that agents only build subagents which achieve a verison of the goal function where the ordinals are decreased. Descents from infinite ordinals may be arbitrarily large, but all such descents must be finite (by the well-ordering of the ordinals).

This has the advantage that the human programmers are not "special" in that they subdivide the goal and the agent merely achieves the goal. By assigning ordinals to subgoals, the agent can divide the goal however it pleases (so long as the ordinals decrease). In this way, the humans do not possess powers which the agent knows not: rather, they merely sit one level higher than the agent on the ordinal stack.

The actual techinque will be outlined in another paper, and will not be described in detail here. A similar technique will be used in Fallenstein's Monster, but it is different enough that we delay explaining the technique until later. For now it suffices to understand that Parametric Polymorphism can subvert the procrastination paradox by means of indexing subgoals with ordinals.

6. Baby Abomination

Marcello's Waterfall lives in a nonstandard universe and succumbs to the procrastination paradox, but it does not suffer from any losses of proof strength (that we yet know of). Parametric Polymorphism has standard models and avoids procrastination, but loses proof strength as its goals draw nearer. We present a third tiling agent by grafting Marcello's Waterfall onto Parametric Polymorphism in a way that achieves most of the best of both systems.

First, we will coerce Marcello's Waterfall into having nice behavior by putting it into a world with (finite but not necessarily discrete) timesteps.

We begin by noting that, though Marcello's Waterfall is unsound, it is "Π_1 sound". If a Π_1 statement is not true then there is a Σ_1 statement exhibiting its falsity which Marcello's Waterfull can find. Because the waterfall is consistent, it cannot prove false Π_1 statements,

and is thus sound for Π_1 sentences. (Furthermore, a result from the December workshop described in a note parallel to this one describes how a sound version of the waterfall can be constructed in which each level merely shows the Π_1 instances of the soundness schema for the next level, rather than all instances, which is sufficient for our purposes here.)

Second, we subdivide the goal G into $G(t)$, where t is a timestep.

Now consider an agent with the following condition for action:

Proposal 3. $\bar{a} \implies MW \vdash \bar{a} \rightarrow \forall t : G(t)$

This agent only executes action a in the case that Marcello's Waterfall (MW) derives that the execution of a results in the goal being achieved on each timestep. Thus, for example, the goal "press the button" might be expressed as

Example 4. $G(t) = t > 7 \rightarrow \exists u < t : \overline{b(u)}$

In other words, "After time seven, there is a proof that at some time before t the button was pressed".

In order for this to work, we require that there is only a finite amount of computation allowed per timestep. (Otherwise, MW could find a proof of inconsistency in ZF and use it to exhibit a nonstandard proof that its successor will halt and press the button.) Assuming ZF is consistent, MW can find no such proof with finite computation, and thus MW can only find trustworthy proofs that there is a time when the button is pressed.

This is a tiling agent that avoids the procrastination paradox so long as humans have subdivided the goal. This is not a particularly satisfactory solution, as you can hardly say you've solved the procrastination paradox when you change the goal from "press the button in some timestep" to "press the button before this timestep".

We can do better.

7. The Full Monster

Before we exhibit the full monster, we must introduce some new notation and make more explicit this idea of "subdividing our goals".

Firstly, we divide our goals into discretely many subgoals $Goals$, which we denote by g. (Thus, we will quantify like $\forall g \in Goals$.) We assume $Goals$ is countable and that we have a fixed enumeration of the goals.

Secondly, we assign each subgoal an ordinal describing the "postponabilitiy" of the goal. Intuitively, this describes the successor depth allowed when achieving the subgoal. It may, however, be an infinite

ordinal, thus allowing goals to be postponed for arbitrarily long (but ultimately finite) amounts of time. We will write this as $Ord(g)$. We require that the range of Ord is restricted to countable ordinals, for simplicity. We also require that the function Ord is "nice" in the following way:

Let U denote the Grothendeick universe corresponding to the smallest inaccessible cardinal κ. We require that Ord must be absolute for U. Formally, $ZFC \vdash \forall \alpha < \kappa : Ord(\alpha) \leftrightarrow Ord^U(\alpha)$, where Ord^U is obtained by replacing all quantifiers $\forall x$ and $\exists x$ by $\forall x \in U$ and $\exists x \in U$. Intuitively, this requires that the behavior of Ord does not depend upon the inaccessible cardinals in the universe.

Next, we introduce a predicate $Achieves$ which takes a goal and a natural number and indicates whether the given goal has been achieved at the given timestep.

Then we introduce a predicate $Inacc(\alpha)$ which states that there exists some strictly increasing function from α onto inaccessible cardinals.

For example, when g is the goal requiring that the button be pressed, we might have the following function expressing that g has been achieved at time t:

Example 5. $Achieves(g, t) = \exists n < t : \overline{b(n)}$

Finally, we are going to require that the agent not only execute an action a, but exhibit a program c. This program takes goals as input and gives as output natural numbers. This program will not actually be run, but the agent must be able to prove that it halts when run on any given goal.

We are now ready to graft a new arm onto the monster. Here is our condition for action:

Proposal 6. $\overline{(a, c)} \implies MW \vdash \bar{a} \rightarrow \forall g : Achieves(g, c(g))$
$\wedge ZFC + Inacc \vdash \forall g \in Goals : Inacc(Ord(g)) \rightarrow Halts(c(g))$

Translated, the agent is licensed to execute action a and exhibit program c whenever it meets the following two conditions:

(1) Marcello's waterfall proves actions are only taken when each goal g is achieved within $c(g)$ timeusteps
(2) $ZFC + Inacc$ proves that for each goal g, if there are indeed $Ord(g)$ inaccessible cardinals then $c(g)$ halts.

Assuming $c(g)$ halts at some standard time, it is easy to see that this agent will take actions only when it knows that taking the actions will cause the goals to be achieved before some standard time. Notice that the proof that $c(g)$ halts is done in ZFC, not the unsound MW.

Let's take a look at how such agents can tile. Intuitively, a parent agent A_0 can only license the construction of a child agent A_1 when the parent trusts the child's proofs that the goals will be achieved *and* when the parent trusts that the child's counter program will halt.

A_0 is able to achieve trust that the goals will be achieved via Marcello's Waterfall. Furthermore, A_0 can only conclude $Inacc(Ord(g)) \rightarrow Halts(c_1(g))$ when it trusts its child's proof of the same. This trust can only be achieved when the child exhibits a proof of $Innac(\beta) \rightarrow Halts(c_1(g))$ for $\beta < Ord(g)$.

This follows from the fact that for all ϕ absolute for U,

Statement 7. $ZFC \vdash \Box_{ZFC} \forall n \in \mathbb{N} : \ulcorner \forall \alpha : Inacc(\alpha) \rightarrow \phi(\bar{n}) \urcorner \rightarrow \forall \alpha : Inacc(\alpha^+) \rightarrow \phi(n)$

Intuitively, ZFC can only "unwrap" statements of the form $\Box \ulcorner Inacc(\beta) \rightarrow \phi \urcorner$ into $Inacc(\alpha) \rightarrow \phi$ when $\beta < \alpha$. Thus, agents using the above condition for action only trust successor agents that prove they halt using strictly less strength. By the well-ordering of ordinals, we conclude that successor chains must be finite (even if they may be arbitrarily large).

8. Extensions

Such an agent can be extended to allow its successors to make observation by allowing the program c to accept as input the observation vector, so long as it can exhibit a proof that c halts on all possible inputs.

9. Discussion

This system (Fallenstein's Monster) has the same strength at all timesteps when reasoning about which actions achieve which goals. At the same time, it only approves of successor agent chains that use progressively less power to prove that they will eventually achieve the goals, in a manner that guarantees all goals will be achieved in some finite amount of time.

Also, the nonstandardness of Marcello's Waterfall is largely mitigated by requiring that the agent prove a Π_1 statement and may only use finite computation. Furthermore, we only need the soundness schema to hold on Π_1 formulas, which means that we can in fact use the sound version of Marcello's waterfall described in a parallel note.

Overall, this agent combines most of the best properties of both Marcello's Waterfall and Parametric Polymorphism.

Note that this agent is still not particularly pleasant (hence the name). We do not recommend building such an agent: we merely note that this agent is a technical improvement over both Parametric Polymorphism and Marcello's Waterfall.