Logical AI Software Specification & Reasoning: GSSOTC

Tau.net

Ohad Asor

October 2, 2024

This is an extended abstract, a two-pager for the logician. For more information cf. the preprint "Guarded Successor: A Novel Temporal Logic". The methods described here are being protected by a patent application.

1 NSO: Nullary Second Order Logic

The goal of NSO is to have a language that can speak about its own sentences in a consistent and decidable way. Tarski's "Undefinability of Truth" has shown that this is impossible under a certain broad setting. The key of NSO is to abstract sentences, so much so, that they make merely Boolean algebra (BA) elements. In particular, there is no access to the syntax of the sentences (in contrast to Tarski's setting).

Any classical logic closed under Boolean combinations makes a BA, called the Lindenbaum-Tarski algebra (LTA) of that logic. Recall that this is only up to logical equivalence. Now observe two important points: 1. Any such logic that has an infinite signature (whether constant, relation, or function symbols), makes an atomless BA. 2. All countable atomless BAs are isomorphic (which is a well known theorem)[1]. Clearly all sentences in languages of interest are finite strings made of finite alphabet, hence countable. The countable atomless BA is therefore the LTA of major logics of interest.

When we say "the theory of BA interpreted in a fixed BA $\mathcal{B}$" we mean not only the first order theory of BA interpreted in $\mathcal{B}$ (recall that an interpretation is a mapping taking symbols from the signature to actual objects in a structure), but we also mean that its signature is equipped with constants that are interpreted in each element of $\mathcal{B}$, so each element has a unique constant assigned to it. We will refer to those constants as the *interpreted constants*. Now fix a language $\mathcal{L}$ that its LTA makes a countable atomless BA. Consider NSO[$\mathcal{L}$] to be the theory of BA interpreted in that LTA, so each sentence in $\mathcal{L}$ is a constant symbol in NSO[$\mathcal{L}$]. So far, NSO[$\mathcal{L}$] is a language that speaks about $\mathcal{L}$, but still not about itself. To achieve that, we make the LTA of NSO[$\mathcal{L}$] to be an atomless BA as well (as currently it is only the two-element BA, as any logic that is interpreted in a fixed structure). This can be done by adding infinitely many uninterpreted constant symbols (the *uninterpreted constants*), or any other such trick. Now the interpreted constants can be extended to include sentences in NSO[$\mathcal{L}$]. Since both $\mathcal{L}$ and NSO[$\mathcal{L}$] make a countable atomless BA, they are isomorphic. By that, and by a few more technical details, we can make NSO[$\mathcal{L}$] speak (including quantify) over its own sentences. Further, NSO[$\mathcal{L}$] is decidable iff $\mathcal{L}$ is decidable.

2 Guarded Successor

Traditional computation is temporal manipulation of bits. Bits, are the elements of the smallest possible Boolean algebra. The construction here can be seen as a generalization of this into working over certain infinite Boolean algebras. Decidability of a specification language in this model is of

[1] Moreover, all atomless BAs are elementarily equivalent, as proved by Tarski.

course much less trivial. Further, we will show how this generalization can support some very surprising abilities.

GSSOTC stands for Guarded Successor Second Order Time Compatible. Here we will not deal with the "second order" part although it is pretty much implied from the setting described here. As an intuitive starting point, any formula with two free variables, in any logic, can be seen as defining a set of sequences: $\phi(x,y)$ can be seen as defining a set of sequenecs such that a sequence is in the set, iff any two consecutive elements x,y satisfy $\phi(x,y)$. Now consider the class of logics having the following property: fix a finite set of constant and variable symbols. Then the number of formulas making use only of those constant and [free] variable symbols (we allow arbitrary quantified variables), up to logical equivalence, is finite. The most relevant such logic is the theory of atomless BA. Let's call it here the "finiteness property" (though in other documents we use the term "weakly ω-categorical theories"). We say that a sequence s models $\phi(x,y)$ and write $s \models \phi$, iff $\phi(s_{n-1}, s_n)$ holds for all n. Denote by $|s|$ the length of s.

Given $\phi(x,y)$, consider the following process: ask whether exists s s.t. $|s| = 2$ and $s \models \phi$, then whether exists s s.t. $|s| = 3$ and $s \models \phi$, and so on. This series of questions look like $\phi_n(x) = \exists y.\phi_{n-1}(y) \wedge \phi(x,y)$ where $\phi_n(x)$ means "exists a sequence of length n starting with x", and then to get a final answer we of course need to consider $\exists x.\phi_n(x)$. Due to the finiteness property, this series of questions is going to loop, and even reach a fixed point due to the monotonic nature of the setting. We obtain a result of the form: "if a sequence of length N exists, then a sequence of any larger length exists". It is easy to see that this also implies the existence of an infinite sequence.

For now we mention only three additional points:

1. Seen as program specification language, those sequences are actually "outputs" or "states", however we'd like to support "time-compatible inputs" as well. This means that we'd like to prove that for each input, at each point of time, exists an output, that does not depend on future inputs (*time-compatible*). So we can deal with formulas of the form $\phi(x_n, x_{n-1}, y_n, y_{n-1})$ where x_n, x_{n-1} are the current and previous inputs, respectively, and similarly for y_n, y_{n-1} for outputs (observe the *bounded lookback* in every formula). The quantifier pattern would look like $\forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots$. It is easy to express it as a recurrence relation as above, and again use the finiteness property. We obtain the following very rare properties of the temporal logic GS: it is a temporal logic that operates over an infinite domain of inputs and outputs, where this domain is equipped not only with equality but also with the theory of atomless BA, and we can prove that for all input exists a time-compatible output.

2. We have to justify the name GS (Guarded Successor). It so happens that the free-variable formulation is equivalent to adding a sort of natural numbers, and function symbols $\mathbb{N} \to \mathcal{D}$ where $\mathcal{D}$ is the domain of the original language, and where the successor relation may appear only in a guarded fashion. for example, $\phi(x_n, x_{n-1}, y_n, y_{n-1})$ would be written as

$$\forall nk.s(n,k) \to \phi(x(n), x(k), y(n), y(k))$$

   where s is the successor relation. This equivalence is only for the universal fragment of the guarded successor logic, however by direct means we can support any quantifier pattern. It can easily be shown that we can support arbitrary quantification over the natural numbers, e.g. $\forall n \exists k \forall m$ and in fact in the GS setting they collapse to a single quantifier (exercise). It is also possible to have a more complex guard. The precise definition is: the guard should allow to determine the exact relative distance between the time points. Supporting the resulting combination of purely universal with purely existential formulas requires a bit more work.

3. Combining NSO with GS gives us a software specification language, with the above desirable properties, and where inputs and outputs, may be sentences in this very same language. This can, for the first time, support implementations of the form: "reject a software update if it doesn't satisfy certain desired properties". It is therefore a crucial ingredient in AI safety.