Diasynchronic Logic

Justin Brody JUSTIN.BRODY@FANDM.EDU
Don Perlis PERLIS@CS.UMD.EDU

Abstract

The primary contribution of this paper is that we sketch a logic which will model a single logical sentence as unfolding over time in concert with meta-awareness. This allows us to implement meta-control of a single sentence; allows for meaningful (and non-paradoxical) self-reference in a single sentence; and employs real time self-reference and control to model thought processes that modify themselves as they unfold.

1. Introduction

In this paper we want to model a logical reasoning agent which thinks a single thought in any given moment. This has some immediate consequences. First, of all the possible consequences of its current knowledge base, it will have to make some kind of choice about what the next thought is. This is in contrast to traditional reasoning engines, which do not instantiate a single thought at a time. Thus, in modelling everday thinking, we are already led to some kind of need to control the thinking process (by choosing the next thought). From this, one might be very quickly led to a view of real-world logical reasoning in which thoughts are instantiated one at a time while some secondary system controls the flow of thoughts.

In this paper, we present a logical framework for modelling both parts of such reasoning. Our framework will work within the context of time-situated reasoning, so that logical sentences are instantiated at particular timesteps of the reasoning process. More specifically, our framework will work on two timescales. The control process will operate timestep by timestep, at what we dub the *synchronic* level. Each individual "thought" will unfold over a sequence of timesteps unified into a single "moment", at what we dub the *diachronic* level. The interactions between these two levels, which will give the theory it's power, will be called the *diasynchronic* level and gives the framework its name of *Diasynchronic Logic*, heceforth abbreviated as DSL when convenient.

Envisioning our framework raises a number of immediate questions, which we touch on here before providing a more complete discussion at the end of this paper. First, one might ask about the connection between the thought instantiated at moment m and that at moment $m + 1$. In traditional logic it is natural to conceive of a growing knowledge base as mediated by logical entailment, or at least logical consistency. That is, we imagine growing a knowledge base via the application of sound inference rules, or possibly also by the addition of observational data which can be conceived as logically neutral. Indeed, this is how the knowledge base grows (in the absence of contradictions) in an active logic agent.

We imagine something similar happening at the diachronic level of our system, where a thought at moment m is often a logical consequence of the thoughts that came before it. There are some noteworthy exceptions, however. In the first place, unlike in active logic agents we will not instantiate all immediate consequences of our knowledge base at once. Such consequences will rather become *potential inferences*, available to be drawn when appropriate. By instantiating consequences one at a time we display fidelity to the singular nature of human trains of thought and also importantly mitigate the notorious combinatorial explosion of inferences which plagues traditional logical systems. This comes at a cost however, in that the mechanism which chooses which potential inference will be drawn at a given moment must be designed

carefully, and indeed one could easily lose desired logical properties like completeness with an poorly chosen mechanism.

As in active logic, we might also grow our knowledge base by means of observations. Further, we might allow for growth in the knowledge base through inductive and abductive reasoning, which in our current framework might be performed at the synchronic timescale and hence look like extra-logical information from the diachronic point of view.

One might summarize these observations by noting that the picture thus far is of a logic which, at the diachronic level, is radically incomplete. By this we do not mean that all the logical consequences of a knowledge might not be derived (although that is true too); we also mean that the contents of the diachronic knowledge base at moment m do not determine the knowledge base at time $m + 1$. Specifically, the logic which operates at the diachronic level is far from a deterministic machine, there are a number of factors that control the operation of this level (i.e. synchronic reasoning) which are not diachronically visible.

Our fundamental view, then, is that thinking is a process which unfolds over time and is subject to meta-cognitive control in much the same way that a servo-ing process can involve motion, cognition and perception in a tightly coupled loop. Analogously to servoing, the unfolding of these sentences will be subject to cognitive observation and control, so that the manner in which a sentence unfolds is not determined *a priori* but is rather subject to modification based on evolving circumstances. The diachronic level is a form of *thick time* in which single moments of cognition are conceived as occuring over an interval rather than in an instant. This also echoes William James' notion of a *specious present* [9] and Husserl's model of time-consciousness in which the present is determined by forces of *retention* and *protention* [22].

The primary contribution of this paper is to give a model of a control process which allows for single logical sentences to unfold over time, in concert with meta-awareness. As a result

1. We describe how to implement meta-control of a single sentence.

2. We give a non-arbitrary account of how a sequences of synchronic instances can be "self-unified" into a single moment. Here, "self-unification" means that the synchronic processes determine the moment boundaries are not somehow imposed from outside of the process.

3. We allow for meaningful (and non-paradoxical) self-reference in a single sentence. For example, we will model instantiations of sentences like "This sentence has 5 words" or "This thought is taking too long".

4. We employ real time self-reference and control to model thought processes that modify themselves as they unfold. For example, an agent counting people entering a room to see if there are too many might have a thought akin to "I see 8, no 9, no 10 people – this is taking too long – let's say at least 10 people". Such a thought can then be used in further reasoning (e.g. "if there are at least 10 people, the room is over capacity").

5. As with human agents, but unlike in most reasoning systems, we model a rational process in which one sentence is processed at a time, and in which some kind of rational process can be defined which determines the flow of such thoughts.

It is worth emphasizing some of the ways in which DSL differs from classical logic. In classical logic (e.g. a forward chainer), a new sentence is produced as an automatic consequence of the information in an agent's knowledge base. In DSL, new sentences are produced more intentionally – of the massive number of sentences which might be logical consequences of a knowledge base, a new sentence that is produced is constructed based on some notion of utility (to include veracity, but also possibly notions of efficiency and effectiveness). Further, the production of new sentences is subject to self-modification.

We note some of the basic ways in which the mechanisms introduced here might be used; we will flesh these out in greater detail in Section 6.

1. Agents can correct their thought process. In mid-thought the facts may seem to change and so the agent may produce a different sentence from the one it was originally working on. For example, in counting the number of people in a room an agent might start to instantiate a sentence "there are 5 people in the room". In the middle of this process, however, more people trickle in. Rather than completing the original sentence and trying to retract it, a DSL agent will be able to modify the sentence in mid-production so that it becomes something more like "there are 5 people – make that 7 people – in the room".

2. A DSL reasoning process might keep a placeholder that will be filled in pending a computation. For example, the agent might produce a sentence "there are [n] words in this sentence" and fill in the correct value of $[n]$ at an appropriate time. Once such a sentence is finalized, it can become a part of the reasoning process.

3. A sentence can evolve in relation to it's own unfolding, as in a thought ". . . this sentence is getting way too long!" that begins but needs to be cut short or another "I was speaking in French but now I'll switch to English." in which a realization causes the unfolding of an utterance to change mid-utterance.

2. Related Work

This work grows out of a larger program to come to terms with the notion of a "processual self", as described in [4]. The latter paper examined the role of a self as a fundamental concept that unifies disparate cognitive phenomena. It also examined the notion of processes able to modify themselves in a single moment of "thick time"; the current work attempts to realize this idea and thus represents a preliminary step in a full accounting of self.

Many of the ideas in [4] were also expressed in a paper of John Barnden's [3]. One point emphasized by Barnden was the role of top-down and bottom-up causality in self-processes. It turns out that Varela and Maturana had tackled similar issues in biology, and also learned of their solution which they termed *autopoiesis* [11], [21]. The theory of autopoeisis ("self-creation") describes a process by which sub-cellular processes unite to form a single cell; the cell itself is both created by the sub-cellular processes and has a causal effect on them. This is all mediated by the cell membrane. As Christian Coseru [5] has pointed out, the membrane thus serves to eliminate the need for notions of downward causality by allowing the membrane to act as a nexus for mediating between bottom-up and top-down processing. This idea led us to imagine the temporal endpoints of sentences as potentially playing a similar role in logical processing.

This work is also an attempt to approach what Perlis has deemed *strong self-reference* [17] and draws philosophically on enactive ideas of unifying action and perception [14], [15]. It is also worth pointing out a connection with Jim Reggia's notion of a *computational explanatory gap*: in [19] Reggia has argued that understanding how neural processes give rise to symbolic reasoning has the potential to go a long way toward understanding the mechanisms of consciousness. The present work can be viewed as attacking the same gap in understanding from the top down (as opposed to from the bottom up).

We view temporality and specifically processing that can be tied to the current moment as fundamentally important. A current text on planning [8] describes a similar approach in planning ("temporal planning"). One might, in some sense, view the present paper as unifying ideas in temporal planning with capabilities for logical representation and inference.

Along those lines, a paper of Douglas Appelt's [2] discusses planning utterances and details a number of useful axioms for various natural-language scenarios. His system plans full utterances before speaking them; but we hope to incorporate some of his insights in future implementations of the system described in this paper.

Finally, this work is itself at the end of a long line of work in active logic. Perhaps the closest work in this history is Madhura Nirkhe's doctoral thesis [13]. In this work, Nirkhe discussed planning processes

which took account of themselves in meeting deadlines. Michael Miller's dissertation [12] was concerned with enable agents to reason about the past to change their beliefs; in some ways this paper can be seen as extending his ideas to allow reasoning about the present and the future. Much of the initial groundwork for active logic was laid out in [6] and a preliminary semantics was developed in [1].

3. Syntax

3.1 Active Logic

To our knowledge, the primary example of a logic which models reasoning occurring at specific times (an *internal logic*, as in [16]) is Active Logic, developed by the Active Logic, Metacomputation and Mind group at the University of Maryland, although Pei Wang's Non-Axiomatic Reasoning System [23] also has some of these features. We will employ active logic as a foundation for developing DSL, our plan is basically to have an active logic agent produce tokens in an "object language" and introduce unifying mechanisms to bind temporal streams of tokens into sentences.

We will assume for the remainder of this paper that the reader has basic familiarity with first-order logic (see, e.g., [7] for an introduction). Syntactically, an active logic sentence is a simply a sentence in a first-order language that has a unary predicate Now[1]. This keeps track of a notion of the *current* time step, and is part of what distinguishes active logic as a diachronic (as opposed to simply temporal) logic: reasoning is situated in a particular time.

Definition 3.1. Let L be a first-order language (that is, L is a finite set of predicate and function symbols of specified arity, with "=" being implictly counted amongst the binary predicates). Then L is an *active logic language* if L contains the unary predicate symbol Now.

Note that a language is purely lexical – it defines the symbols that will be used in a logical formalism. The meaning of those symbols comes from the set of axioms expressed in that language and any inference rules that are defined in the logic. In particular, the meaning of Now is determined by the time update rules given in Figure 4.1.1.

We think of active logic agents as evolving over time. Specifically an active logic knowledge base $\mathbf{K}_0$ is initialized at time 0 and for any natural number t, the knowledge base at time t (denoted $\mathbf{K}_t$) comprises the agent's knowledge base at time t. If $\sigma \in \mathbf{K}_t$, we denote this fact by writing $t : \sigma$.

There are two primary drivers of change between $\mathbf{K}_t$ and $\mathbf{K}_{t+1}$. In the first place, $\mathbf{K}_t$ can grow from deduction. Specifically the *modus ponens* inference rule states that if ϕ and $\phi \rightarrow \psi$ are both in $\mathbf{K}_t$, then we will have $\psi \in \mathbf{K}_{t+1}$. In Section 4.4.1.1 we discuss the inference rules which can change the knowledge base going from time t to time $t + 1$. It is important to note that the *modus ponens* inference rule only adds sentences which are derivable from $\mathbf{K}_t$ by a **single** application of *modus ponens* (with time-stamps replaced appropriately). In particular an agent will not immediately know all the logical consequences of its knowledge base, even if all such consequences will be derived at some point in time. Thus logical consequence is not an instantaneous relation but a process that unfolds over time.

There are also non-deductive ways in which information can enter the knowledge base; specifically an agent can make observations which provide knowledge that is not deduced from axioms. We will see examples in which such observations enter the knowledge base in Section 6.

One of the distinguishing properties of active logic is that it enables reasoning that takes place at a particular time and can be *about* a partiuclar time. In particular, for any time t we will have $\mathrm{Now}(t) \in \mathbf{K}_t$; intuitively this means that an agent always has access to the current timestep.

1. A fuller version of active logic will also include predicates for belief and contradiction; we omit these here for simplicity. See [1] and [10] for more detail.

We give a simple example to illustrate the workings of active logic. Suppose that at time 0 the sentences $p, p \to q, q \to r$ are added in the knowledge base $\mathbf{K}_0$. Thus $\mathbf{K}_0$ will consist of the active logic sentences $\{ \mathrm{Now}(0), p, p \to q, q \to r \}$. Applying the time update rule and modus ponens rule adds $\mathrm{Now}(1)$ and q in the next timestep, while an inheritance rule maintains the presence of the non-temporal sentences from $\mathbf{K}_0$. Note specifically that r is not a one-step consequence of $\mathbf{K}_0$ even though it is a logical consequence: it takes two applications of modus ponens to deduce r from $\mathbf{K}_0$. Thus we will have $\mathbf{K}_1 = \{ \mathrm{Now}(1), p, p \to q, q \to r, q \}$ and $\mathbf{K}_2 = \{ \mathrm{Now}(2), p, p \to q, q \to r, q, r \}$.

A final necessary feature for our active logic basis will be the existence of a quoting mechanism. In particular, we will assume that for any first order language L that we work with, if σ is an L-sentence, then so is $\ulcorner \sigma \urcorner$. This will require a certain amount of machinery to realize, which we omit in the present discussion – see [1] or [7] for possible approaches.

There is much more that could be said about active logic. For example a preliminary model theory is worked out in [1]; an omitted feature of the logic is that it is paraconsistent and contains contradiction handling mechanisms [18], [12], and there is a working implementation, the Active Logic MAchine [18]). Perhaps the most detailed formal treatment of the syntax is in [10]. The interested reader is referred to these papers.

3.2 DSL Syntax

To model diasynchronic logic, we will work with two languages: a meta-language L_S and an object language L_D. We think of L_S as controlling operations at the synchronic level while L_D describes the diachronic level.

The object language L_D will be a first-order language. Since our goal is to have L_D sentences dynamically constructed as quoted entities in L_S, we will need tokens which will be the building blocks of such entities. Thus we associate with L_D a finite set of tokens $\mathbf{T}_D$ which have the property that for any L_D sentences σ, we can write σ as a concatenation of tokens from $\mathbf{T}_D$. For technical convenience we will insist that that empty string ε is an element of $\mathbf{T}_D$. In addition, predicates $\mathrm{thisMoment}, \mathrm{msentence}, \kappa$ will be present, along with an operator $\diamond$ – these all correspond to the L_S predicates defined below.

We define our terms formally. First, a diachronic language is defined as follows.

Definition 3.2. A *diachronic (logical) language* $(L_D, \mathbf{T}_D)$ consists of a first-order language L_D and a set of tokens $\mathbf{T}_D$ such that:

1. Every sentence of L_D can be written as a finite concatenation of elements from $\mathbf{T}_D$.
2. $\varepsilon \in \mathbf{T}_D$. That is, the empty string is a token.
3. L_D contains constants symbols $\{ [n] : n \in \mathbf{N} \}$. These will serve as placeholder tokens which will be replaced with the results of computations.
4. L_D contains a predicate $\diamond(\ulcorner \sigma \urcorner)$ indicating that the semantic content of the current diachronic sentence from before the current time should be replaced with σ.
5. L_D contains a unary predicates `thisMoment(m)` indicating the current moment number is m.
6. L_D contains a function `msentence(m)` indicating the L_D sentence at moment m.

A slight variant on the notion of a diachronic language allows us to model utterances instead of logical reasoning.

Definition 3.3. A *diachronic natural language* $(\varnothing, \mathbf{T}_N)$ consists of a set of tokens $\mathbf{T}_N$ with $\varepsilon \in \mathbf{T}_N$.

The basic difference is that in a diachronic logical language well-formed sentences will be produced, with a logical relation between sentences at different moments. In a diachronic natural language we simply produce strings with no inherent relation amongst the strings produced at different moments.

Given such a language, a synchronic language will be an active logic language in which we can control the production of L_D sentences.

Definition 3.4. Given a diachronic language $(L_D, \mathbf{T}_D)$, the triple $(L_D, L_S, \mathbf{T}_D)$ is a diasynchronic language if:

1. L_S is an active logic language.
2. L_S contains unary predicates $\beta(s), \gamma(t)$. These will denote the respective beginning and end of a diachronic sentence.
3. L_S contains a binary predicate $\alpha(T_0, t)$. This will be used to denote the instantiation of a token from L_D at a particular time. In particular, $\alpha(\ulcorner T \urcorner, t)$ denotes that the token T occurs at synchronic time t.
4. L_S contain a predicate `thisMoment(m)` indicating the current moment number is m.
5. L_S contains functions `msentence(m)`, `tsentence(t)` referring to the sentence at moment m or time t, respectively.
6. L_S contains a binary computation predicates $\kappa([n], \ulcorner f \urcorner)$. For some token-valued function f, the predicate $\kappa([n], \ulcorner f \urcorner)$ indicates that upon completion of f, occurences of placeholder $[n]$ in the current L_D sentence will be replaced with the token which is the output of f.

If $(L_D, L_S, \mathbf{T}_D)$ is a diasynchronic language, we call L_S a *synchronic language*. Note that this defintion still holds when the diachronic language is natural, in which case we will call $(L_D, \varnothing, \mathbf{T}_N)$ a natural diasynchronic language.

We conceive of a DSL agent as instantiating (at most) a single token of the object language at each time step. Specifically, the predicate $\alpha(\ulcorner T \urcorner, t)$ will be used to indicate that the token T in the object loangauge is instatiated at synchronic time t (the quotes $\ulcorner \cdot \urcorner$ are for Gödel-style coding; we will use the regular quotes " " when the tokens are natural language words). Note that this formula can appear at any time; we imagine the actual instantiation as occurring at time t. This is just a conceptual convenience; there is no actual instantiation occurring except for the formulae in the meta-language[2]. For $s < t < u$, $s : \alpha(\ulcorner T \urcorner, t)$ can be interpreted as the agent's *intention* to instatiate T at time t while $u : \alpha(\ulcorner T \urcorner, t)$ will indicate the agent's *recollection* that T was instantiated at time t, while $t : \alpha(\ulcorner T \urcorner, t)$ represents the actual instantiation of T at time t.

At any given time step, it is assumed that some diachronic sentence (either a logical sentence or an utterance) is currently underway. The first time step of the current sentence is marked by $\beta(s)$ and the last is denoted by $\gamma(t)$. In particular, if some sentence is undertaken between time step s and t, then for $s \leq t_0 \leq t$, we will have $t_0 : \beta(s)$ in the knowledge base. We will *not* necessarily have $t_0 : \gamma(t)$ in the knowledge base, since we want to allow for the possibility that $\gamma(t)$ is determined (and updated) in the middle of a sentence production. For any time t, $\Sigma(t)$ will denote the set $\{ (s, T) : \beta(t) \leq s \leq \gamma(t), \alpha(\ulcorner T \urcorner, s) \in \mathbf{K} \}$ where $\mathbf{K}$ denotes the current knowledge base.

As a syntactic convenience, we refer to the DSL sentence at moment 0 as the concatenation of all tokens T where T is instantiated between synchronic time 0 and synchronic time $\gamma(0)$. Note that until the sentence is complete (marked by the first synchronic instantiation $t_0 : \gamma(t_0)$), the sentence at moment 0 is a dynamic entity. Note also that $\gamma(t)$ is highly indexical until the first moment has passed. We denote the DSL sentence at moment 0 (as it exists at time s) by σ_0^s. We define σ_m^s for $m > 0$ in the analogous manner.

2. In case L_D is a natural language, this could be coupled with a separate process (perhaps one monitoring the knowledge base) to physically utter the token.

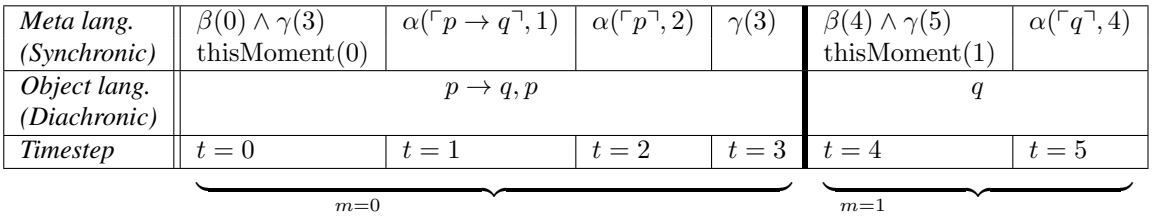

Figure 1. The evolution of a DSL reasoner. The production of sentences in the synchronic language is determined by the L_S-knowledge base (not fully shown) and the inference rules of active logic. The production of sentences in the diachronic language is determined by the instantiations of α in the synchronic language.

4. Inference rules

Since control of the deductive process is a primary characteristic of diasynchronic logic, we want to give DSL agents some control over which sentences it instantiates. Indeed this is somewhat necessary – if an agent has numerous deductions available to it at any given moment but can only reason with one deduction at a time (at a given moment), then it will need some mechanism to decide which one to use. We thus introduce a *pre-inferential buffer* which stores *potential inferences*. Specifically, for any L_D sentence σ, we let $t : \Pi(\sigma)$ denote that at synchronic time t, σ is a potential inference which is available for instantiation. For a diachronic moment m, $m : \Pi(\sigma)$ will be shorthand for $\forall t, moment(m, t) \to \Pi(\sigma)$, where $moment(m, t)$ is shorthand for $\mathrm{Now}(t) \to \mathrm{thisMoment}(m)$ and indicates that synchronic time t occurs in diachronic moment m.

4.1 Synchronic Inference Rules

4.1.1 Active Logic Inference

These include the classical active logic inference rules; see, for example, [1]. Some of the primary active logic inference rules (besides the time update rules 4.1.1) which we will use in our examples are:

- *Time updating inference rule*

$$0 : \mathrm{Now}(0) \tag{1}$$

$$\frac{t : \mathrm{Now}(t)}{t + 1 : \mathrm{Now}(t + 1)} \tag{2}$$

- *Direct contradiction inference rule:* If a contradiction is discovered in the knowledge base, it will be noted in the knowledge base.

$$\frac{t : \phi, \neg\phi}{t + 1 : \mathrm{Contra}(\phi, t)}$$

  There are additional mechanisms to defuse and resolve contradictions so that they do not corrupt the knowledge base; we will not discuss these here but refer the interested reader to the active logic literature (e.g. [6], [10]).

- *Inheritance inference rule:* With the exceptions of the $\mathrm{Now}, \gamma, \beta$ predicates and contradictory sentences, the knowledge base is maintained over time.

$$\frac{t:\phi}{t+1:\phi} \tag{3}$$

for $\phi \neq \text{Now}(t), \phi \neq \gamma(s), \phi \neq \beta(s)$ and $\neg\phi$ not in the knowledge base at time t

- *Sentence delimeter update rules:* Unless a new sentence is started, β will propagate. We allow γ to change without engendering a contradiction.

$$\frac{t:\beta(s), \neg\gamma(t)}{t+1:\beta(s)} \tag{4}$$

$$\frac{t:\gamma(t)}{t+1:\beta(s+1)} \tag{5}$$

- *Modus ponens inference rule:*

$$\frac{t:\phi \rightarrow \psi, \phi}{t+1:\psi} \tag{6}$$

- *Resolution inference rule:*

$$\frac{t: l_1 \vee \ldots \vee l_k, m_1 \vee \ldots \vee m_n}{t+1: \text{SUBST}(\theta, l_1 \vee \ldots \vee l_{k-1}, m_1 \vee \ldots \vee m_{n-1})} \tag{7}$$

where $\text{UNIFY}(l_k, \neg m_n) = \theta$. See Section 9.5 of [20] for details.

4.1.2 Additional Synchronic Rules

In addition, we will have synchronic inference rules that specify that any function calls invoked by κ must complete before a sentence ends. Recall that our syntax allows for the introduction of computation predicates $\kappa([n], \ulcorner f \urcorner)$ for some function f. Ultimately, this should be modeled by some version of λ-calculus which keeps track of the number of timesteps it's using. Since this would take us out of the scope of this paper, we will instead conceive of it as a kind of external function call, so that f will be bound to some function in an external language which replaces all instances of $[n]$ with its return value upon completion. Our assumption is that f has access to everything in the knowledge base at the time it is called; in future work we will replace this with a λ-calculus based formalism[3]

We also want a rule that ensures that our reasoner uses complete sentences. In particular, no free variables should remain in a DSL sentence at the time of it's completion. Thus we have rules of the form

$$\frac{t:\alpha(\ulcorner O([n]) \urcorner, s), \gamma(t+1)}{t+1:\gamma(t+2)} \tag{8}$$

where O is any L_D-predicate. These indicate that a moment should not be completed before function calls has been completed. One result of this rule is that if no κ predicate calling on a function to replace a placeholder $[n]$ exists within a moment, that moment will never end.

3. It is interesting to note that κ in acts as a kind of quantifier, in the sense that it binds a kind of variable. Future work will explore scoping rules and the effect of alternating κ with $\forall$ and $\exists$

4.2 Diachronic Inference Rules

Our diachronic inference rules both use functions; our assumption is that these will be completed between time-steps and are hence acceptable as part of an inference rule. Because we do not have to keep track of the timing of these functions, there is no need to bind the symbols to external code and we can assume that these are defined in L_S.

Our basic diachronic inference rule will be of the form

$$\frac{m : A, \phi(\ulcorner A \urcorner, k) \rightarrow B}{m : \Pi(B)} \tag{9}$$

where ϕ is a predicate representing a computable function of A; i.e. $\phi(\ulcorner A \urcorner, k)$ holds if and only if $f_\phi(\ulcorner A \urcorner) = k$ for some function f_ϕ. This will allow statements like "If at moment m the tail end of A indicates that there are 12 people in the room and B indicates that more than 10 people in the room implies that the room is over capacity, then I can legitimately conclude that the room is over capacity." Note that the potential inference is added at the same moment even though it can only be instantiated in the next moment.

In order to select the next instantiated DSL-sentence, we also have a rule of the form

$$\frac{m : \Pi(B), i(\ulcorner B \urcorner)}{m + 1 : B} \tag{10}$$

that is, if B is a legitimate conclusion and i is a predicate indicating that B is preferred, then B will be produced in the next moment. We can think of the function i as imposing an ordering on the set of potential sentences.

4.3 Diasynchronic Inference Rules

To instantiate rule (10) we introduce some machinery:

$$\frac{t : \text{thisMoment}(m), \Pi(B), i(\ulcorner B \urcorner)}{t + 1 : \tau(\ulcorner B \urcorner, m + 1)} \tag{11}$$

indicating that we intend to instantiate B at moment $m + 1$.

To actually do the instantiation:

$$\frac{t : \text{thisMoment}(m + 1), \beta(t), \tau(\ulcorner B \urcorner, m + 1)}{\alpha(\text{tok}(j, \ulcorner B \urcorner), t + j)} \tag{12}$$

where $\text{tok}(j, \ulcorner B \urcorner)$ is a function returning the jth token of B.

We also have a moment updating rule

$$\frac{t : \text{thisMoment}(m), \gamma(t)}{t + 1 : \text{thisMoment}(m + 1)} \tag{13}$$

5. Semantics

Since DSL theories are reducible to Active Logic theories, most of the model theory necessary to interpret a DSL sentence σ is that given by the semantics of active logic; see [1] for details. We add some extra provisions for our new basic predicates:

1. For sentences ϕ, ψ, we have $m : \phi \diamond \psi$ if ϕ was in the knowledge base at some synchronic time during m and ψ is true at the end of the moment m.

2. The predicate $t : \text{thisMoment}(m)$ is true just in case there are exactly m predicates of the form $s : \gamma(s)$ with $s < t$.

3. For any synchronic time t, define

$$\delta(t) = \begin{cases} T & \text{if for some } T, t : \alpha(\ulcorner T \urcorner, t) \in \mathbf{K}_t \\ \varepsilon & \text{otherwise} \end{cases} \tag{14}$$

   For any m, the set of mth DSL tokens is the set $T_m := \{ (\delta(j), j) : j : \text{thisMoment}(m) \}$; with the set of mth DSL timestamps being given by $S_m := \{ j : \exists(a, j) \in T_m \}$. Then the function $\text{msentence}(m)$ is interpreted as the concatentation $\prod_{j=\min(S_m)}^{\max(S_m)} \delta(j)$ and `tsentence(t)` is interpreted as `msentence(m)` where $t : \text{thisMoment}(m)$ (note that m is well-defined).

4. $t : \kappa([n], \ulcorner f \urcorner)$ if at time t function f has been called and if complete, all occurences of $[n]$ in the knowledge base have been replaced with the returned result of f.

It is worth emphasizing that many DSL sentences are assertions not just about the state of the world but also about an agent's mental state. Specifically, the agent and its knowledge base are included in the model theory. This essential characteristic of an internal logic [16] is covered in detail in [1].

In future work, we will fully work out the model theory associated with DSL. As noted, the basic defintiions are determined by their status as Active Logic theories, but we will explore specific phenomena that arise from the extra structure imposed on them by DSL.

6. Examples

6.1 "This sentence lasts 3 timesteps."

We demonstrate the production of an object-level sentence which counts the number of timesteps it has. In this example, suppose our length computation f takes 2 timesteps (in reality it would probably be much quicker) and returns $t - s$ where $\gamma(t), \beta(s)$ both hold. Here L_D has the extra predicates $\text{numsteps}(m, n)$ which indicates that the number of steps at moment m is n. Notice that everything takes place in a single moment.

Unless otherwise noted, each formula is propagated from synchronic time t to time $t + 1$ (by the Inheritance Rule of Active Logic).

Timestep	Sentence	Notes
$0:$	$\beta(0)$	
	$\text{thisMoment}(m) \rightarrow \alpha(\ulcorner \text{numsteps}(m, [0]) \urcorner, 0)$	
	$\kappa([0], f)$	Use f to determine length
	$\gamma(1)$	
$1:$	$\gamma(2)$	By Inference Rule (8)
	$\alpha(\ulcorner \text{numsteps}(0, [0]) \urcorner, 0)$	
$2:$	$\gamma(3)$	
	$\alpha(\ulcorner \text{numsteps}(0, 3) \urcorner, 0)$	
3	$\gamma(3)$	

It is worth noting that we refer to "this sentence" by refering to the sentence at the current moment using $\text{thisMoment}(m) \rightarrow \alpha(\ulcorner \text{numsteps}(m, [0]) \urcorner, 0)$. For this example, we would have achienved the same effect by using $\alpha(\ulcorner \text{numsteps}(0, [0]) \urcorner, 0)$, but this is a non-indexical expression which is better understood as "the sentence at moment 0" than "this sentence".

It is also worth pointing out that inference rule (8) ensures that the sentence will not complete until f has resolved the referent of $[0]$.

Diachronically, our example is simply the instantiation of the sentence $0 : \mathrm{numsteps}(0, 3)$ indicating that at moment 0, the sentence instantiated at moment 0 has 3 steps.

6.2 "I count 8, no 9, make that more than 10 people in the room."

The basic idea here is that agent will keep counting while people are entering the room. To avoid endless counting, if there are more than 10 people the agent knows what it needs to know – that the room is over capacity and the counting process can stop.

Timestep	Sentence	Notes
$0:$	$\beta(0), \mathrm{numPeople}(0)$	
a	$\gamma(3)$	Initial γ estimate
b	$\mathrm{newPeople}(k) \wedge \mathrm{numPeople}(n) \wedge \neg\,\mathrm{overCapacity} \rightarrow \mathrm{numPeople}(n+k)$	Track $\mathrm{numPeople}$ in synchronic time
c	$\mathrm{newPeople}(k) \wedge \mathrm{numPeople}(n) \wedge \neg\,\mathrm{overCapacity} \wedge \mathrm{Now}(t) \rightarrow \alpha(\ulcorner\Diamond(\ulcorner\mathrm{numPeople}(n+k)\urcorner)\urcorner, t+1)$	Track $\mathrm{numPeople}$ in diachronic time
d	$\mathrm{numPeople}(n) \wedge n > 10 \rightarrow \mathrm{overCapacity}$	Defining $\mathrm{overCapacity}$
e	$\mathrm{newPeople}(k) \wedge \gamma(t) \wedge \neg\,\mathrm{overCapacity} \rightarrow \gamma(t+2)$	Need extra time
$1:$	$\mathrm{newPeople}(8)$	From observation
	$\alpha(\ulcorner\Diamond(\ulcorner\mathrm{numPeople}(0)\urcorner\urcorner, 1)$	From 0.c
$2:$	$\mathrm{numPeople}(8)$	
	$\alpha(\ulcorner\Diamond(\ulcorner\mathrm{numPeople}(8)\urcorner)\urcorner, 1)$	
	$\gamma(5)$	From 0.e
$3:$	$\mathrm{newPeople}(1)$	
$4:$	$\alpha(\ulcorner\Diamond(\ulcorner\mathrm{numPeople}(9)\urcorner)\urcorner, 2)$	
	$\gamma(7)$	From 3, 0.e
$5:$	$\mathrm{newPeople}(3)$	
$6:$	$\gamma(9)$	
	$\mathrm{numPeople}(12)$	
	$\alpha(\ulcorner\Diamond(\ulcorner\mathrm{numPeople}(12)\urcorner)\urcorner, 6)$	
$7:$	$\mathrm{overCapacity}, \mathrm{newPeople}(10)$	
$8:$	$\gamma(11)$	
	$\mathrm{newPeople}(5)$	
$11:$	$\gamma(11)$	

Note that at steps 6, 7 and 8 new people are observed, but they are ignored because the agent has already concluded that the room is over capacity.

This example corresponds to the following diachronic sentence in the object language:

$$\Diamond\,\mathrm{numPeople}(0)\,\Diamond\,\mathrm{numPeople}(8)\,\Diamond\,\mathrm{numPeople}(9)\,\Diamond\,\mathrm{numPeople}(12)$$

6.3 "I count 8, no 9, make that 12 people in the room".

This example is similar to the last one, except that the conclusion that the room is over capacity is done in the object language.

Let $\phi(\ulcorner\sigma\urcorner, k)$ be a predicate such that $\phi(\ulcorner\sigma\urcorner, 1)$ is true if the last $\mathrm{numPeople}$ instantiation in σ is a number above 12 and $\phi(\ulcorner\sigma\urcorner, 1)$ is true otherwise [4]

Timestep	Sentence	Notes
$0:$	$\beta(0)$	Initialization
	$\gamma(2)$	Initialization
	$\mathrm{numPeople}(0)$	Initialization
	$\mathrm{thisMoment}(0)$	Initialization
	$\alpha(\ulcorner\phi(\ulcorner\sigma\urcorner, 1) \rightarrow \mathrm{overCapacity}\urcorner, 0)$	Define $\mathrm{overCapacity}$
	$\mathrm{newPeople}(k) \wedge \mathrm{numPeople}(n) \rightarrow$ $\mathrm{numPeople}(n+k)$	Add new people to total
	$\mathrm{newPeople}(k) \wedge \gamma(t) \rightarrow \gamma(t+2)$	Still counting, delay end
	$\mathrm{numPeople}(n) \wedge \mathrm{Now}(t) \rightarrow$ $\alpha(\ulcorner\mathrm{numPeople}(n)\urcorner, t+1)$	Keep numpeople in object sentence
$1:$	$\mathrm{newPeople}(20)$	From observation
	$\alpha(\ulcorner\mathrm{numPeople}(0)\urcorner, 1)$	
$2:$	$\mathrm{numPeople}(20)$	
	$\gamma(4)$	
$3:$	$\alpha(\ulcorner\mathrm{numPeople}(20)\urcorner, 1)$	
$4:$	$\gamma(4)$	
	$\tau(\ulcorner\mathrm{overCapacity}\urcorner, m+1)$	Inference rule (11)
$5:$	$\beta(5)$	
	$\mathrm{thisMoment}(1)$	Inference rule (13)
$6:$	$\alpha(\ulcorner\mathrm{overCapacity}\urcorner), 7)$	Inference rule (12)
$7:$	$\alpha(\ulcorner\mathrm{overCapacity}\urcorner), 7)$	Inheritance

At the diachronic level, this corresponds to the following

Moment	Sentence
$0:$	$\phi(\ulcorner\sigma\urcorner, 1) \rightarrow \mathrm{overCapacity}$
	$\mathrm{numPeople}(20)$
$1:$	$\mathrm{overCapacity}$

Specifically, we're trying to instantiate something like: if the final count of people in the room is greater than 10, then conlude we're above capacity. This allows us to conclude we're above capacity from a thought like "I count 8, no 9, make that 12 people in the room".

6.4 "I see 8, no 9, no 10, no 11 – this is getting to long – call it at least 11 people in the room"

This is similar to the previous example, but we will employ self-reference in DSL. Let us suppose that at time 0 the agent has the following sentences in its knowledge base, where $\phi(\ulcorner\sigma\urcorner, 1)$ holds when σ asserts that

4. The functional mechanisms needed to define such a predicate are omitted here and will be addressed in future work.

there are at least 10 people (or more).

Timestep	Sentence	
0 :	$\gamma(t) \wedge t > 20 \rightarrow \text{tooLong}$	Stop counting after 20 steps
	$\text{tooLong} \rightarrow \text{summarize}$	
	$\beta(0)$	Initialization
	$\gamma(2)$	Initialization
	$\text{numPeople}(0)$	Initialization
	$\text{thisMoment}(0)$	Initialization
	$\alpha(\ulcorner\phi(\ulcorner\sigma\urcorner, 1) \rightarrow \text{overCapacity}\urcorner, 0)$	Define overCapacity
	$\text{Now}(t) \wedge \text{newPeople}(k, t) \wedge \text{numPeople}(n) \rightarrow$ $\text{numPeople}(n + k)$	Add new people to total
	$\text{Now}(t) \wedge \text{newPeople}(k, t) \wedge \gamma(t) \rightarrow \gamma(t + 2)$	Still counting, delay end
	$\text{Now}(t) \wedge \text{numPeople}(n) \rightarrow$ $\alpha(\ulcorner\text{numPeople}(n)\urcorner, t + 1)$	Keep numpeople in object sentence
	$\text{summarize} \wedge \text{Now}(t) \wedge \max(k, t) \rightarrow$ $\alpha(\ulcorner\text{atLeast}(k)\urcorner, t + 1)$	
	$\exists s < t\Big(\alpha(\ulcorner\text{numPeople}(k)\urcorner, s) \wedge$	Counted k people at some point
	$\forall s \forall l\ s < t \wedge \alpha(\ulcorner\text{numPeople}(l)\urcorner, s)\Big) \rightarrow \max(k, t)$	Never counted more than k people
1	$\text{newPeople}(8, 1)$	
2	$\text{numPeople}(8)$	
⋮		
10	$\text{newPeople}(9, 10)$	
11	$\text{numPeople}(17)$	
12	$\text{newPeople}(10, 12)$	
13	$\text{numPeople}(27)$	
⋮		
19	$\text{newPeople}(11, 17)$	
20	$\text{numPeople}(38)$	
21		

It subsequenctly sees 8, then 9 then 10, then 11 people. More people trickle in, but it is past 20 timesteps so it summarize the moment with $\text{atLeast}(11)$. Over the next few moments we can then conclude that the room is over capacity.

6.5 This sentence has 5 words, as an utterance.

We note that the production of a sentence in the target language can be thought of as an utterance as easily as a logical sentence. Indeed, for our purposes we can think of an utterance as a special case of a logical sentence in which there are no deductive rules in the object language.

We illustrate this by sketching what might happen as an agent produces the utterance "This sentence has 5 terms.". We note that this situation could trivially be handled by having the agent decide at time 0 that it will utter the complete sentence and then proceed to do so. The power of this formalism, however, is in the ability to model an agent that simply knows that it wants to make a *true self-referential utterance about the number of words in the utterance*. The agent knows about the truth conditions for such an utterance and

perhaps some different syntactic forms such an utterance can take (more on these later) but does not know at the outset exactly what it will look like: it will perhaps speak randomly within the constraints of wanting to make a true self-referential utterance that counts the number of words it utters.

Axiomatically, the system will assume that it utters a single word at every time step. It thus has an axiom $(\gamma(t) \wedge \beta(s)) \rightarrow \nu(t-s)$ where $t : \nu(n)$ indicates that the utterance being undertaken at time t has n words in it. Thus our axiom says that once the beginning and end of an utterance are determined, we know how many words it will have. We then imagine our agent adding the following sentences to its knowledge base at the indicated times (the exact mechanism for adding each of these will be discussed later).

- At time 0, we will add $0 : \alpha(\text{“}This\text{”}, 1)$, as well as $0 : (\gamma(t) \wedge \beta(s)) \rightarrow \nu(t-s)$ and $\beta(1)$. (For readability, we will omit the time-stamps on individual sentences in what follows). Seeing the next few steps, $\alpha(\text{“}sentence\text{”}, 2)$ and $\alpha(\text{“}has\text{”}, 3)$ are also added.
- At time 1, the agent decides that it will end the utterance with “[n] words” where n is the number of words in the current utterance. It also realizes that it will end in 5 time steps. The sentences $\nu(n) \rightarrow \alpha(\sigma(n), 4), \mathrm{Now}(t) \rightarrow \gamma(t+4), \alpha(\text{“}words.\text{”}, 5)$ are all added. Here $\sigma(n)$ is a function which “converts” the integer to a string.
- At time 2, $\gamma(6)$ will be derived from the penultimate sentence at time 1. We also delete $\mathrm{Now}(t) \rightarrow \gamma(t+2)$ from the knowledge base (we could accomplish the same thing by introducing the notion of a “single-use” inference rule, or writing the implication in a more complicated way that would effectively make it single use (by instantiating some predicate during the current utterance for example). For simplicity here, we imagine just removing it.
- At time 3,$\nu(5)$ will be deriveed from our original axiom.
- At time 4, $\alpha(\text{“}5\text{”}, 4)$ will be added.

6.6 A math problem

Let us imagine an agent working through a calculus problem, perhaps an integration by parts where some function has to be written as a product of two other functions v and du. While it is working through the initial steps of noting that v and du must multiply to give the original function, it can use an external function call to determine some properties of potential choices of v and du. Ideally this can allow it to look forward and make a good choice initially, rather than guessing and backtracking repeatedly.

7. Discussion

This papers gives an initial sketch of a logical formalism for a kind of self-modifying sentence. There is a fair amount of additional work to do, including:

1. Filling in gaps in the presentation so far (e.g. being more explicit about the quoting mechanism)
2. Exploring the model theory associated with DSL.
3. Implementing DSL agents in a computer.

At first sight the machinery presented in this paper may seems fairly heavy and appear unwarranted for cases of simple deduction. However, very straightforward deduction can proceed with very minimal overhead with the simple production of object level sentences and minimal reasoning about them. Indeed it is trivial to use DSL as a simple first-order reasoner in the object language. The benefit of DSL comes about in situations

where control of the reasoning process can be exerted to gain efficiency. In general this will apply to any situation in which a search through a tree of deductive can be made more efficient by implementing a guided search rather than a blind one.

One particular source of such efficiencies comes from the changing nature of an agent's environment. Much of the machinery of mathematical logic was developed in the context of contemplating timeless truths that were not tied to the particular circumstances of the reasoner. For example an agent reasoning about geometry can work within a complete axiomatic system. Agents in the real world are not so fortunate – they always work with incomplete theories and need to be informed by an ever-fluctuating sea of observational data. Indeed for a DSL agent such changes can come both from observing the world and from the addition of new deduced information. Thus such agents need to be aware of their resources and use them efficiently; in particular working with a complete but computationally inefficient reasoning system is insufficient – the answers the agent needs must be timely. Thus an agent must take some control of its own reasoning and in particular avoid having to backtrack in its reasoning process.

We note that as given, DSL is unlikely to be either sounds or complete. Indeed, from the point of view of diachronic sentences, there is a lot of action behind the scenes that is invisible from the at the object level. It is reasonable to expect that L_S by itself is sound and complete and we will attempt to demonstrate this formally in future work.

References

[1] Michael L Anderson, Walid Gomaa, John Grant, and Don Perlis. Active logic semantics for a single agent in a static world. *Artificial Intelligence*, 172(8):1045–1063, 2008.

[2] Douglas E Appelt. Planning natural-language utterances. In *AAAI*, pages 59–62, 1982.

[3] John Barnden. Running into consciousness. *Journal of Consciousness Studies*, 21(5-6):33–56, 2014.

[4] Justin Brody, Michael T Cox, and Donald Perlis. The processual self as cognitive unifier. In *Proceedings of the Annual Meeting of the International Association for Computing and Philosophy*, 2013.

[5] Christian Coseru. *Perceiving reality: Consciousness, intentionality, and cognition in Buddhist philosophy*. Oxford University Press, 2015.

[6] Jennifer Jill Elgot-Drapkin. Step-logic: reasoning situated in time. 1988.

[7] Herbert B Enderton. *A mathematical introduction to logic*. Academic press, 2001.

[8] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016.

[9] William James. The perception of time. *The Journal of speculative philosophy*, 20(4):374–407, 1886.

[10] Darsana Purushothaman Josyula. *A unified theory of acting and agency for a universal interfacing agent*. PhD thesis, Department of Computer Science, University of Maryland, College Park, Maryland, 2005.

[11] Humberto R Maturana and Francisco J Varela. *Autopoiesis and cognition: The realization of the living*, volume 42. Springer Science & Business Media, 1991.

[12] Michael J Miller. A view of one's past and other aspects of reasoned change in belief. 1993.

[13] Madhura Nirkhe. *Time-situated reasoning within tight deadlines and realistic space and computation bounds*. PhD thesis, research directed by Dept. of Electrical Engineering.University of Maryland at College Park, 1994.

[14] Alva Noë. *Action in perception*. MIT press, 2004.

[15] J Kevin O'Regan and Alva Noë. A sensorimotor account of vision and visual consciousness. *Behavioral and brain sciences*, 24(05):939–973, 2001.

[16] Don Perlis, Justin Brody, Sarit Kraus, and Michael J Miller. The internal reasoning of robots. In *COMMONSENSE*, 2017.

[17] Donald Perlis. Consciousness as self-function. *Journal of Consciousness Studies*, 4(5-6):509–525, 1997.

[18] Khemdut Purang. *Systems that detect and repair their own mistakes*. PhD thesis, Department of Computer Science, University of Maryland, College Park, Maryland, 2001.

[19] James A Reggia, Derek Monner, and Jared Sylvester. The computational explanatory gap. *Journal of Consciousness Studies*, 21(9-10):153–178, 2014.

[20] Stuart J Russell and Peter Norvig. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[21] Evan Thompson. *Mind in life: Biology, phenomenology, and the sciences of mind*. Harvard University Press, 2007.

[22] Francisco J Varela. The specious present: A neurophenomenology of time consciousness. 1999.

[23] Pei Wang. Non-axiomatic reasoning system (version 4.1). In *AAAI/IAAI*, pages 1135–1136, 2000.