NOTES ON TIME IN AGI

Mike Archbold
Northwest AGI Forum
February 2022

SUMMARY

Here are some notes that can be used in conjunction with the Philosophy Club's event about time, (https://www.meetup.com/The-Philosophy-Club/events/283053894/), a very general exploration of the philosophy of time. The meetup is not an AGI meetup per se, but I put together some notes and links regarding the status of time theory in some of the more prominent, open AGI architectures for our edification and discussion.

The notes herein are not comprehensive in nature but intended only to convey some general feel for the present state of time theory in AGI practice. Just to emphasize – the coverage below is uneven and only intended to give a general feel and some links for consideration, and not a critical overview in any way. Thanks to all who responded to my inquiries. Most of this info was cut/pasted from Wikipedia, the projects' websites, and emails/message boards. I tried to attribute appropriately.

CONTENTS

1) OPENCOG has an “active agent… able to use temporal reasoning” although in “exploratory phase”

Overview: https://wiki.opencog.org/w/CogPrime_Overview

A FEW GENERAL COMMENTS FROM THE DEVELOPERS FOLLOW

Ben Goertzel
Feb 2, 2022, 1:57:49 PM (yesterday)
to opencog, Nil

Nil is doing a lot right now with temporal reasoning in PLN for learning/planning in Minecraft. So temporal representation is an active area….

Nils Geisweiller
Feb 2, 2022, 9:19:25 PM (17 hours ago)
to Ben Goertzel, opencog

Yeah, Hedra and I are actively working on temporal reasoning, the documentation is not well centralized at this point, and we're in exploratory phase but with functioning code and an agent able to do planning in uncertain/unknown environments using temporal/procedural reasoning (it uses temporal pattern mining as well, but that can be seen as specialized form of temporal reasoning).

I've one comment about time and temporal theory, and I suspect you'll hate it. I strongly believe that an AGI system must learn about time, (and how to reason about events in time) instead of having it hard-coded into it. Likewise, it must learn "common sense", and only after that, can it learn about rationality and reasoning. Thus, I spend all my effort on learning; I expect it to (eventually) learn common sense and how to reason about time. This is why I (personally) don't work on any explicit theories of time.

This differs from older (dare I say "conventional"?) architectures, where all this stuff (space, time, reasoning, logic) is hard-coded in at a base layer.

-- **Linas Vepstas**

It's (above) probably true in theory, though in practice it helps to have readily available temporal representations/rules/procedures. **Nil (Geisweiller**)

LINKS FOR OPENCOG TIME THEORY

thread discussion → https://groups.google.com/g/opencog/c/BL0CpIh3fDg

- Presentation at AGI-21

https://odysee.com/@ngeiswei:d/AGI-21-Temporal-Procedural-Reasoning-Nil-Geisweiller-Hedra-Yusuf:6

- PLN book (chapt 14) http://goertzel.org/PLN_BOOK_6_27_08.pdf

- Wiki page https://wiki.opencog.org/w/Category:Temporal_Reasoning

- PLN repo https://github.com/opencog/pln (see for the currently

implemented temporal rules and their documentation

https://github.com/opencog/pln/tree/master/opencog/pln/rules/temporal)

- ROCCA repo (reasoning-based agent control in unknown environments)

https://github.com/opencog/rocca

~~~~~~~~~~~~~~~~~

The present research and development centers for time around the use of temporal predicates in PLN. The basic goal of PLN is to provide reasonably accurate probabilistic inference in a way that is compatible with both Term Logic and Predicate Logic, and scales up to operate in real time on large dynamic knowledge bases. PLN is able to encompass within uncertain logic such ideas as induction, abduction, analogy, fuzziness and speculation, and **reasoning about time** and causality. **PLN** is a novel conceptual, mathematical and computational approach to uncertain inference. In order to carry out effective reasoning in real-world circumstances, AI software must robustly handle uncertainty. The Unified Rule does forward and backward chaining, looking like CLIPS a bit to me.

SOME ADDITIONAL COMMENTS FROM LINAS FOLLOW:

(About the chaining engine looking a bit like CLIPS):
And not by accident. There are, however, some deep and fundamental differences. These are:

* The "rules" are kept in a graph database that can be saved to disk in several formats, saved to SQL, no-SQL, and transmitted by network to other network nodes.

* The graph store is more generic than just "rules", you can store anything you want in it. It's a generalized KR system. If you don't like the default KR style, you can invent your own: all knowledge graphs are not just static graphs, but are also executable, and you get to pick how that's done. (OK, so if you invent your own, it might not work so well with PLN, and whatever temporal subsystem gets created. So compatibility is your responsibility, too.)

* Unlike CLIPS (or Prolog) rules/expressions can have more than just true/false values. They can be given floating-point valuations, for example, Bayesian probabilities or fuzzy-logic percentages. They can be given vector-of-floats, e.g. two numbers: probability & confidence. Or a vector of 653 floats, from some neural net. Or a vector of strings. Or a nested tree of floats and strings. Or whatever. Each valuation is a generic key-value DB. And not just only "true/false".
~~~~~~~~~~~~~~~~~

The default PLN rules that are CLIPS-like use a blend of probability theory and fuzzy logic. But again, you don't have to use these, you can invent your own.

LINAS COMMENTS ABOUT NARS, SOAR, ACT-R

I want to draw a few more distinctions. First, "classic" OpenCog is (was?) a theory of mind or a theory of cognition (a "cognitive model"?), having more than a few similarities to the above systems. This "classic" OpenCog is described in several books by Goertzel et al, and assorted papers, conference proceedings, etc. Assorted variants of it were built.

All of these incarnations of OpenCog were built on a generic infrastructure, the "Atomspace". The AtomSpace is meant to provide an "easy-to-use" base on which different cognitive theories can be created, explored, developed. It tries to be impartial, providing a collection of tinker-toy parts which you can assemble yourself, or extend, implement, re-implement as needed to pursue any one particular theory or vision of what cognition is.

Because we've turned the crank on this 3 or 4 or 5 times, the lower layers have gotten fairly generic, and are debugged, stable, performance-optimized and can support the weight of more complex devices to be built on top of them. The exploration of higher layers continues unabated. Most of what you abstracted (ie., the following notes) about NARS, SOAR, ACT-R would count as "higher layers".

To rephrase: the AtomSpace allows you to "roll your own" temporal logic. I don't care- have at it, use your favorite theory. You mention ACT-R as having declarative, and procedural memory, and ACT-R being a production system. Sure, we can do all three styles in the AtomSpace, simultaneously, on the same data. I don't care: do it however you want. You bolded: At each moment, an internal pattern matcher [in ACT-R] searches for a production that matches the current state of the buffers. Only one such production can be executed at a given moment By contrast, in the AtomSpace, you can run productions one at a time, or in parallel, or distributed across the network. Don't care. Or, instead of productions, you can use term-rewriting, graph rewriting, don't care. The toolset is there.

2) NARS “there are *two* primitive temporal relations: “before” and “when”.

Overview of NARS: https://cis.temple.edu/~pwang/NARS-Intro.html

The following information was copied from the NARS documentation:

The truth-value of a statement in NARS is time-dependent in principle, as conceptual relations change over time. However, for efficiency and implementation consideration, it is not always necessary to take the temporal attribution of a statement into account, so, by default, truth-values are not marked with temporal information, and are treated as eternal. Whenever the temporal attribute of a truth-value needs to be specified, the statement is considered an event. Therefore event is not listed in the grammar as a separate category.

An event is a statement with a time-dependent truth value, that is, the evidential support summarized in its truth-value is valid only in its duration, which is a certain period of time from the moment the event starts to the moment it ends. Thus, an event is a statement whose truth-value may change, and this type of change is different from the changes caused by the accumulation of evidence. During the specified time period of an event, the evidence collected before the period becomes outdated.

In NARS, time can be represented indirectly through events and their temporal relations. The temporal relation between two atomic events E1 and E2 has been designed to fall in one of the following three categories:

1. *E1 happens before E2,*
2. *E1 happens after E2,*
3. *E1 and E2 happen at the same time.*

The first two categories can be expressed using the same temporal relation and therefore, there are two primitive temporal relations: “before” and “when”. “before” relation is irreflexive, antisymmetric, and transitive while “when” is reflexive, symmetric, and transitive.

The following comments are from Patrick Hammer of the NARS project:

“The NARS descriptions (editor: above) in your document regarding time are still valid, my only critique is that they are quite generic. Over the years we have filled in many details we are happy to share with you if you want. We also implemented the principles in an efficient way, and demonstrated them to work well in rich streams of events such as necessary to control a robot with multiple sensor modalities. Getting this right was pretty much our main focus between 2015 and 2020, together with other sensorimotor aspects and attention allocation which highly depends on timing. Since we are very happy with the outcome we moved on to other issues and the strengths of NAL-based declarative reasoning.

Here a very simple example of an a-b event sequence using "OpenNARS for Applications" (https://github.com/opennars/OpenNARS-for-Applications):
Input: a. :|: occurrenceTime=1 Priority=1.000000 Truth: frequency=1.000000, confidence=0.900000
Input: b. :|: occurrenceTime=2 Priority=1.000000 Truth: frequency=1.000000, confidence=0.900000
Derived: dt=1.000000 <a =/> b>. Priority=0.348301 Truth: frequency=1.000000, confidence=0.282230
As you see, there is an occurrence time value which is assigned to each event, this is how before/after can be decided, and how the dt (time delta) of the induced hypothesis is calculated. The time delta is required to decide the occurrence time of a prediction of **b**. Additionally, if **b** does not happen after **a**, negative evidence is attributed to the hypothesis **<a =/> b>**. Additionally, a projection formula is used to decay the confidence of a conclusion dependent on time distance between the premises, but patterns which span higher time distances can still become higher-confident than short time-distance ones via revision (repeated occurrence). Projection also allows to revise hypotheses with varying time deltas to learn proper timing expectations (since timing itself is uncertain), and to handle timing variations in decision making.

In the reasoning literature there are many ideas how to formalize and handle time, but most of them wouldn't work for AGI, or aren't practical for various reasons such as not being able to handle timing variations and uncertainties in timing in general even though they are crucial. As a rule of thumb, I suggest to be skeptical about anything which only exists in papers, it's way easier to describe something than to describe something which could really work, and then to make it work reliably. I'm convinced timing happens to be a key aspect of AGI as I think you have rightly identified, it's one of the things which have to be properly handled at the beginning and is hard to add to a system later. Humans' attention allocation and decision making is strongly bound to the current moment, though our decisions are not fully determined by the current moment but strongly controlled by our intentions and also previous results of reasoning.”

See: https://github.com/opennars/opennars/wiki/Temporal-Inference

3) SOAR “no specific theory of time”

Overview of SOAR, a venerable, widely used cognitive architecture:

> The goal of the Soar project is to develop the fixed computational building blocks necessary for general intelligent agents – agents that can perform a wide range of tasks and encode, use, and learn all types of knowledge to realize the full range of cognitive capabilities found in humans, such as decision making, problem solving, planning, and natural language understanding. It is both a theory of what cognition is and a computational implementation of that theory. Since its beginnings in 1983 as John Laird’s thesis, it has been widely used by AI researchers to create intelligent agents and cognitive models of different aspects of human behavior. The most current and comprehensive description of Soar is the 2012 book, *The Soar Cognitive Architecture. (*Wikipedia)

In the 2012 book above it was remarked that SOAR was in need of a time theory, and this situation is still the case. This seems like an indication of how far a robust cognitive architecture can be developed, over nearly 40 years, performing many functions, *without* a unifying time theory. (me)

Mike,

We do not have a specific theory of time encoded in Soar. For some applications we have used Allen’s temporal logic.

John (Laird)

4) ACT-R – has been used for time research

Overview cobbled from Wikipedia:

ACT-R clearly belongs to the "symbolic" field and is classified as such in standard textbooks and collections. Members of the ACT-R community, including its developers, prefer to think of ACT-R as a general framework that specifies how the brain is organized, and how its organization gives birth to what is perceived (and, in cognitive psychology, investigated) as mind, going beyond the traditional symbolic/connectionist debate. Like a programming language, ACT-R is a framework: for different tasks (e.g., Tower of Hanoi, memory for text or for list of words, language comprehension, communication, aircraft controlling), researchers create "models" (i.e., programs) in ACT-R. These models reflect the modelers' assumptions about the task within the ACT-R view of cognition. The model might then be run. Running a model automatically produces a step-by-step simulation of human behavior which specifies each individual cognitive operation (i.e., memory encoding and retrieval, visual and auditory encoding, motor programming and execution, mental imagery manipulation). ACT-R's most important assumption is that human knowledge can be divided into two irreducible kinds of representations: *declarative* and *procedural*. Within the ACT-R code, declarative knowledge is represented in the form of *chunks*, i.e. vector representations of individual properties, each of them accessible from a labelled slot. Chunks are held and made accessible through *buffers*, which are the front-end of what are *modules*, i.e. specialized and largely independent brain structures. Procedural knowledge is represented in the form of *productions*. The term "production" reflects the actual implementation of ACT-R as a production system, but, in fact, a production is mainly a formal notation to specify the information flow from cortical areas (i.e. the buffers) to the basal ganglia, and back to the cortex. **At each moment, an internal pattern matcher searches for a production that matches the current state of the buffers. Only one such production can be executed at a given moment**. That production, when executed, can modify the buffers and thus change the state of the system. Thus, in ACT-R, cognition unfolds as a succession of production firings.

In a paper titled “Temporal Counting On ACT-R to Represent Time” by Cassenti and Reifers, the authors noted that “**temporal cognition is a field given surprisingly little attention in cognitive modeling.**” Although somewhat dated (2005), it seems like the situation is about the same. (me)

Computational Models of Time Perception

Komala Anamalamudi
Research Scholar at SCIS
University of Hyderabad and
Associate Professor in CSE
Madanapalle Institute of
Technology & Science, Madanapalle
Email: komal_nag@yahoo.com

Bapi Raju Surampudi
Professor at SCIS and
Coordinator for CNCS
University of Hyderabad
Email: bapics@uohyd.ernet.in

Madhavilatha Maganti
DST(WOS-A) Scientist at CNCS
University of Hyderabad
Email: magantimadhavilatha@gmail.com

Excerpts from above 2014 paper (I added emphasis):

"The computational models of time estimation are based on either Symbolic/ Logic based approaches or Connectionist approaches. We observed that **almost all the computational models in time estimation are built on ACT-R** framework which in turn is built on Symbolic/Logic based approach."

The paper outlines some time theories, and reports a variety of research projects, mostly on ACT-R, with a "temporal module" added:
https://www.researchgate.net/publication/268813178_Computational_Models_of_Time_Perception

This webpage provides comprehensive coverage of ACT-R publications:
http://act-r.psy.cmu.edu/publication/ You can search by "time perception."

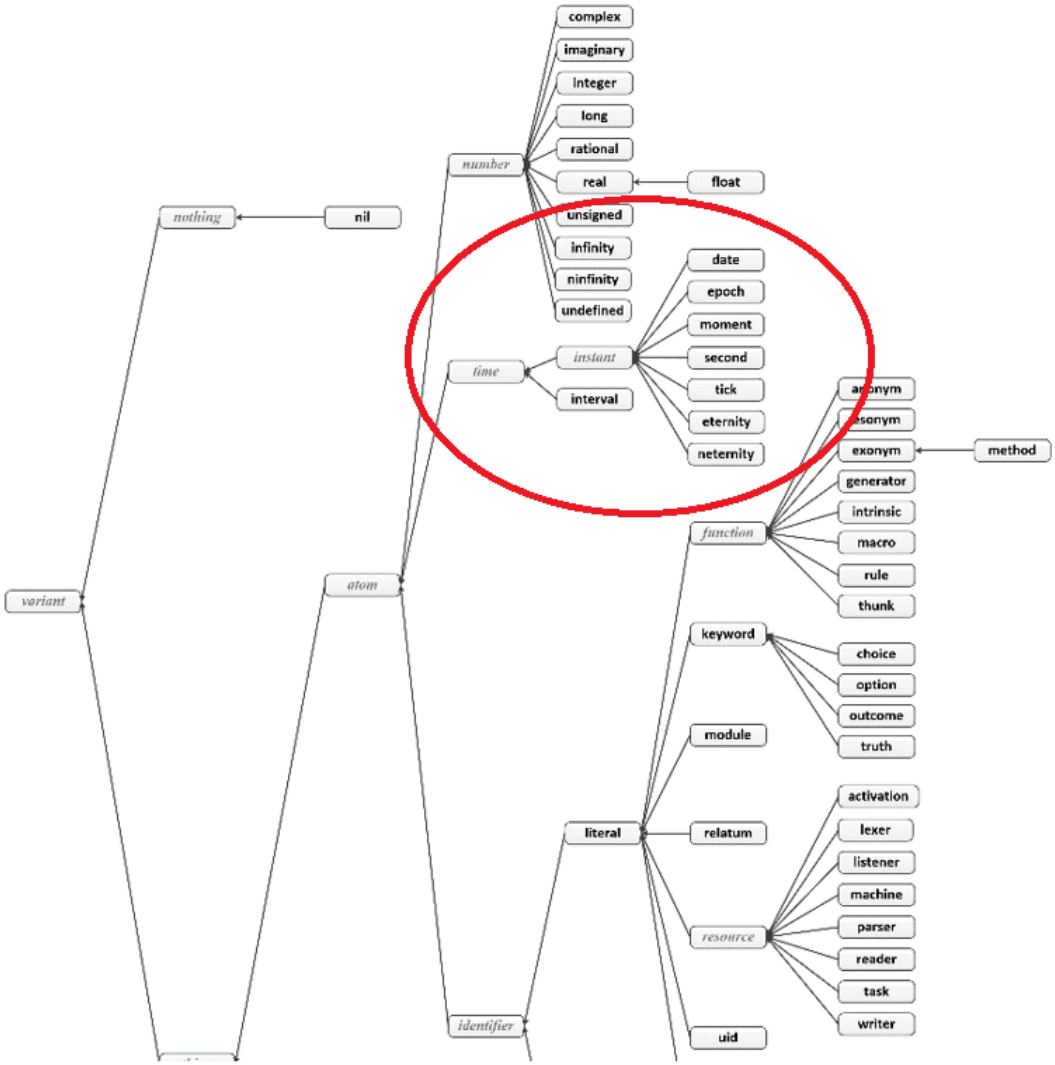
5) PREMISE LANGUAGE

Premise is the work of Michael Miller in the context of his Piagetian Modeler multi-agent cognitive system.

The Premise Language is a functional prototype programming language which combines high level intrinsic primitives with seamless object persistence. The goal of the Premise language is to provide a platform for artificial intelligence programming. Software agents written in Premise share a knowledge base where they can create, modify, and delete object instances amongst themselves in a stigmergic manner. Premise is influenced by Lisp, Self, JavaScript, OPS5, and CLIPS. Premise facilitates declarative, functional, and imperative programming. (from linkedin profile)

The following two images give some basic information of time representation (kind thanks to Michael):

Premise language:



A moment is a nanoyear.
A tick is a nanosecond.

We have functions for comparing time and creating time entities.

before-p

True if an interval finishes before a second interval.

Syntax

(**before-p** *interval$_1$ interval$_2$ tolerance*)

Module

Premise.KB

Parameters

Name	Data Type	Qty	Description
interval$_1$	interval	1	An interval
interval$_2$	interval	1	An interval
tolerance	instant	0-1	Amount of variation. (Defaults to zero ticks).

Results

Data Type	Description
truth	True if interval1 finishes some time before interval2.

Remarks

None.

Example

```
> (before-p  1s|2s  2s|4s)
.: false
> (before-p  1s|5s  6s|8s)
.: true
> (before-p  1s|9s  4s|6s)
.: false
> (before-p  4s|6s  8s|9s)
.: true
> (before-p  4s|6s  6s|9s 1s)
.: false
```

Related

during-p, finishes-p, meets-p, overlaps-p, starts-p

interval functions: before-p, during-p, finishes-p, meets-p, overlaps-p, starts-p.
Instant functions: < , >, = , <=, >=, etc.

TEMPORAL LOGIC APPENDIX NOTES

(Wikipedia / SEP / me mashup follows)

In a temporal logic, a statement can have a truth value that varies in time—in contrast with an atemporal logic, which applies only to statements whose truth values are constant in time. Although Arthur Prior is widely known as a founder of temporal logic, the first formalization of such logic was provided in 1947 by Polish logician, Jerzy Łoś. I (Wikipedia)

Much of the early (ie., ancient) temporal discussion, however, centered around the problem of ***future contingents*****, that is, the question whether statements about future events that are neither necessary nor impossible can have definite truth values.** The most widely known and probably most cited example is the sea-fight scenario discussed by Aristotle in *On Interpretation* (Chapter 9). Aristotle argued that statements such as "There will be a sea-fight tomorrow", as well as the contrary prediction "There will not be a sea-fight tomorrow", do **not hold of necessity and hence lack definite truth values at present**, while conceding that it is necessary that either there will be a sea-fight tomorrow or not. (SEP)

From the early 1950s, Prior introduced and analyzed in detail over more than a decade several different versions of Tense Logic (SEP).

(There are) **TWO** basic types of formal models of time together with some of their pertinent properties: instant-based and interval-based models. (SEP)

Interval-based models of time are useful if ***duration,*** not ***precise instants*** are most relevant. (me)

Interval-based models usually presuppose linear time. Still, they are ontologically richer than instant-based models, as **there are many more possible relationships between time intervals than between time instants.** (SEP) For example, overlap of two events (me).

The basic language of Prior's Tense Logic TL extends the standard propositional language (with atomic propositions and truth-functional connectives) by four temporal operators with intended meaning as follows:

- "It has at some time been the case that …"
- "It will at some time be the case that …"
- "It has always been the case that …"
- "It will always be the case that …"

As in the case of modal logic, the language and semantics of TL can be translated into classical first-order logic…. In fact, in the early days of temporal logic, Prior's approach was perceived as a rival to more conventional approaches using first-order logic. The rivalry between tense logic and first-order logic can be seen as reflecting a fundamental distinction concerning the nature of time, namely the distinction between the A-series and the B-series of time introduced by McTaggart (1908)....The A-series essentially amounts to a characterization of time and temporal order in terms of past, present, and future. The B-series, in contrast, is based on the notions 'earlier' and 'later'. Thus, whereas the A-series presupposes a distinguished present, the B-series involves an overarching, global perspective on time. McTaggart argued that the B-series is insufficient because it lacks an appropriate notion of change, and he rejected the A-series as inconsistent because what is future now will be past, which, according to him, requires that one and the same time instant has incompatible properties. From this he concluded that time is unreal. For a detailed discussion of McTaggart's account and its philosophical relevance, see Ingthorsson (2016) and the entry on time. There is a close correspondence between the A-series of time and tense logic, on the one hand, and between the B-series and first-order logic, on the other. **The most popular and widely used temporal logic in computer science** is the linear time temporal logic LTL, which was proposed in the seminal paper Pnueli (1977), and it was first explicitly axiomatized and studied in Gabbay et al. (1980). In LTL, time is conceived of as a linear, discrete succession of time instants. (SEP)

See this link for details: https://plato.stanford.edu/entries/logic-temporal

~~~~~~~~~~~~~~~~~~~~~~~~

Of course, much work has been done on time-series forecasting in AI.
~~~~~~~~~~~~~~~~~~~~~~~~