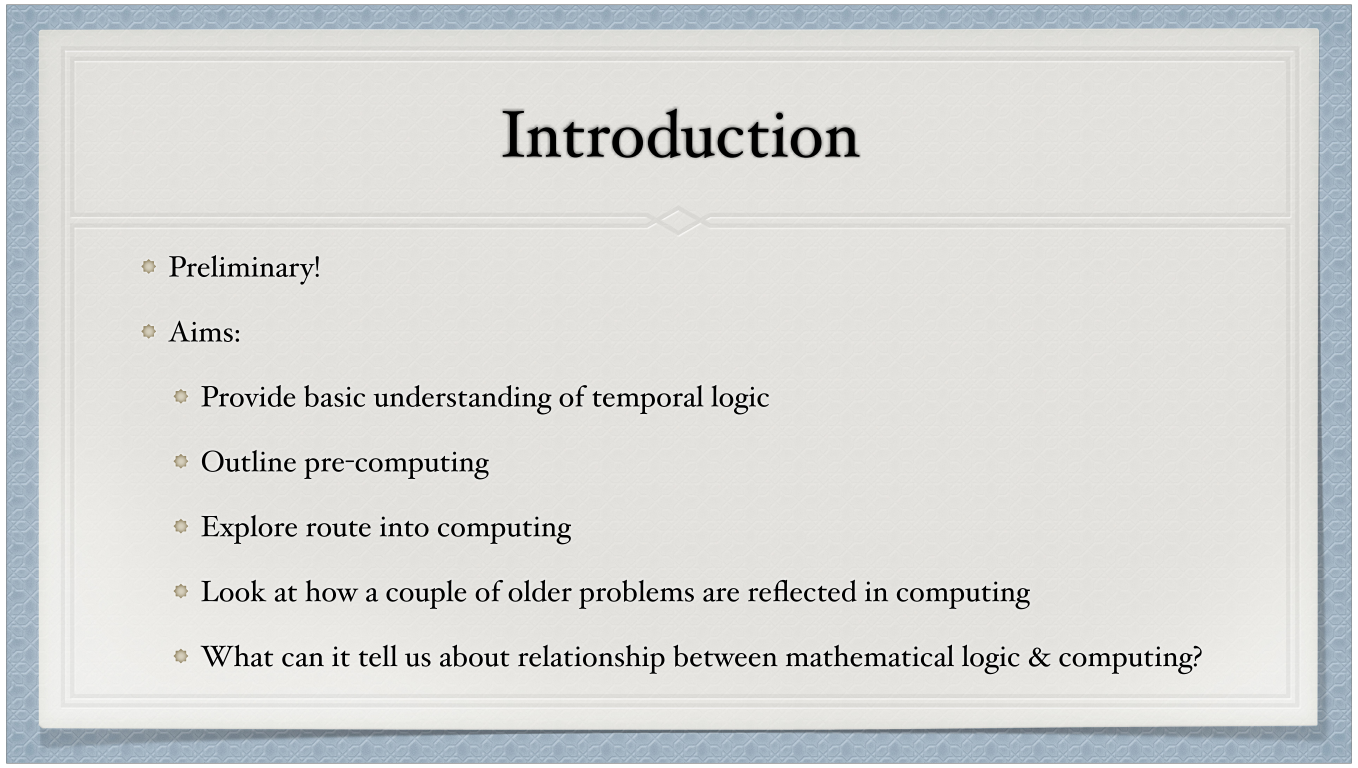
There will be a time-fight tomorrow
Old problems in new logics

Troy Kaighin Astarte
Newcastle University
troy.astarte@ncl.ac.uk

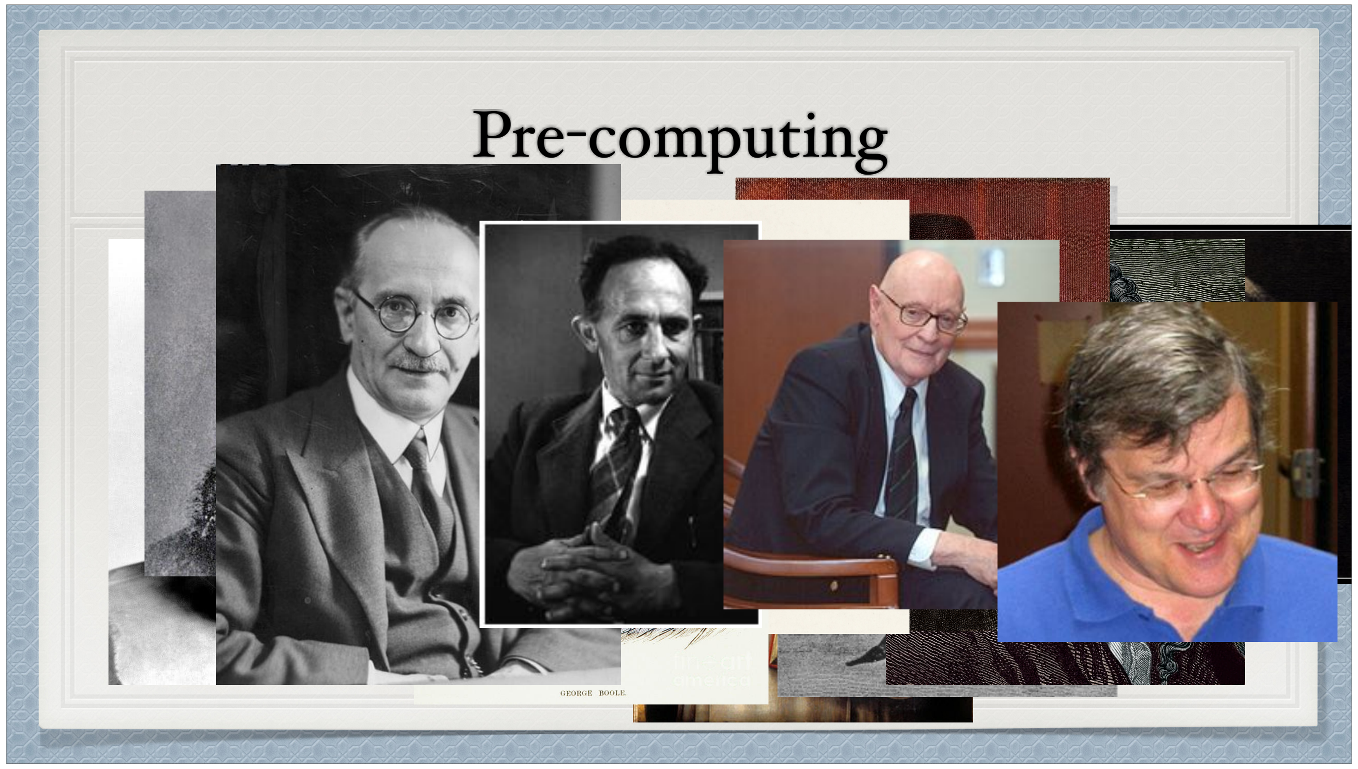
LEVERHULME
TRUST



Introduction

- Preliminary!
- Aims:
  - Provide basic understanding of temporal logic
  - Outline pre-computing
  - Explore route into computing
  - Look at how a couple of older problems are reflected in computing
  - What can it tell us about relationship between mathematical logic & computing?

Rehearsal for ICHST



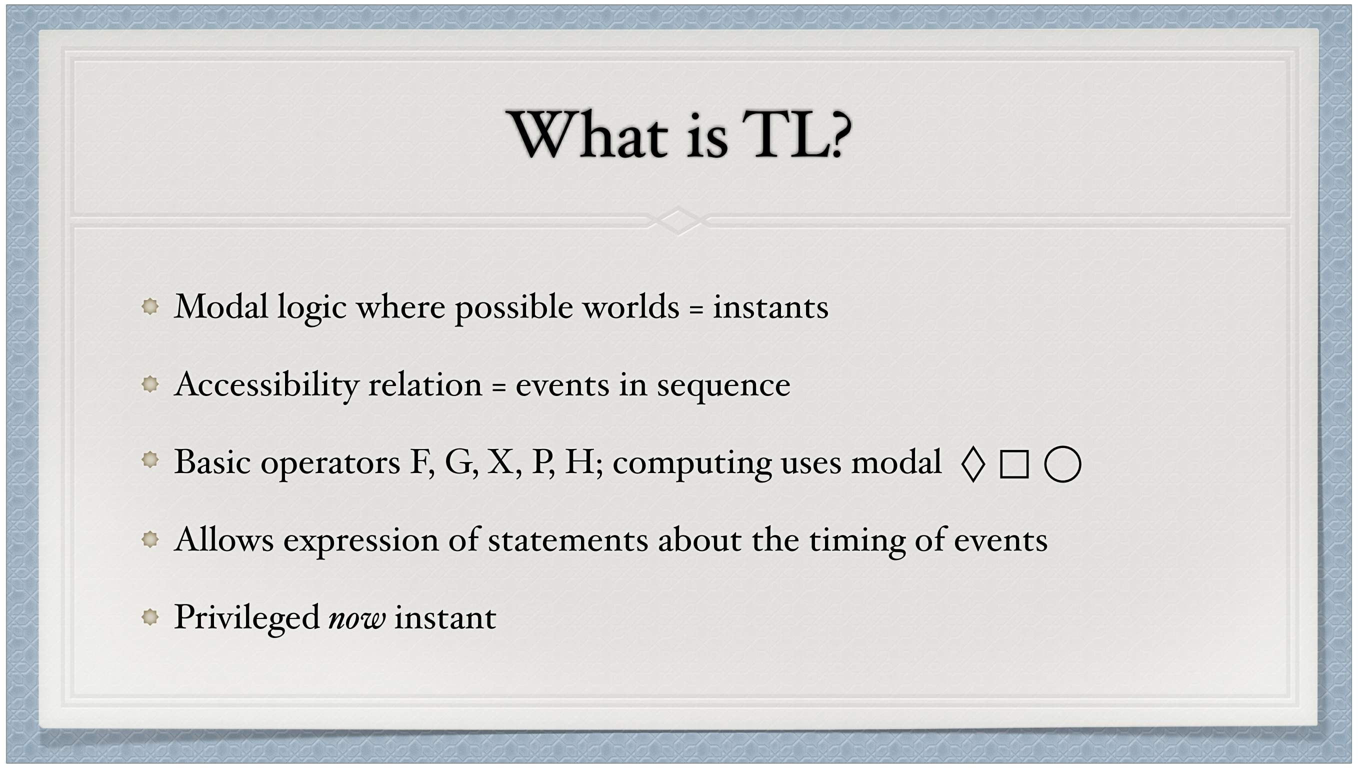
Time in logic has a long history—as far back as Aristotle, who wrote about changing nature of certain statements, and Diodorus, who explored Master Argument about possibility and necessity
Aquinas, Buridan, Ockham serious examination, recognising truth relative to durations being more complex
Natural language contains tensed statements, want to express those; crucial philosophophical link with determinism and omniscience of God (coming later)
though late scholasticism and beginning modern period saw its downfall due to new view of logic as part of the timeless truths such as mathematics, natural philosophy (Leibniz, Bacon)
19th century: Frege, Boole, Peirce included some concern of time, increasingly; Peirce in particular thought about modality and time
Inspired Łukasiewicz's contingent future idea
Which in turn inspired Arthur Norman Prior, (also his Polish notation) who really codified TL
Though it's Rescher and Urquhart's book *Temporal Logic* that finally made the impact on computing

Pre-computing

Arthur Norman Prior (1914–1969)

- Born in New Zealand, raised Methodist
- Ethical logic and history of logic
- Shared mediaeval interest in tense statements
- Locke lectures in 1955–6 led to book

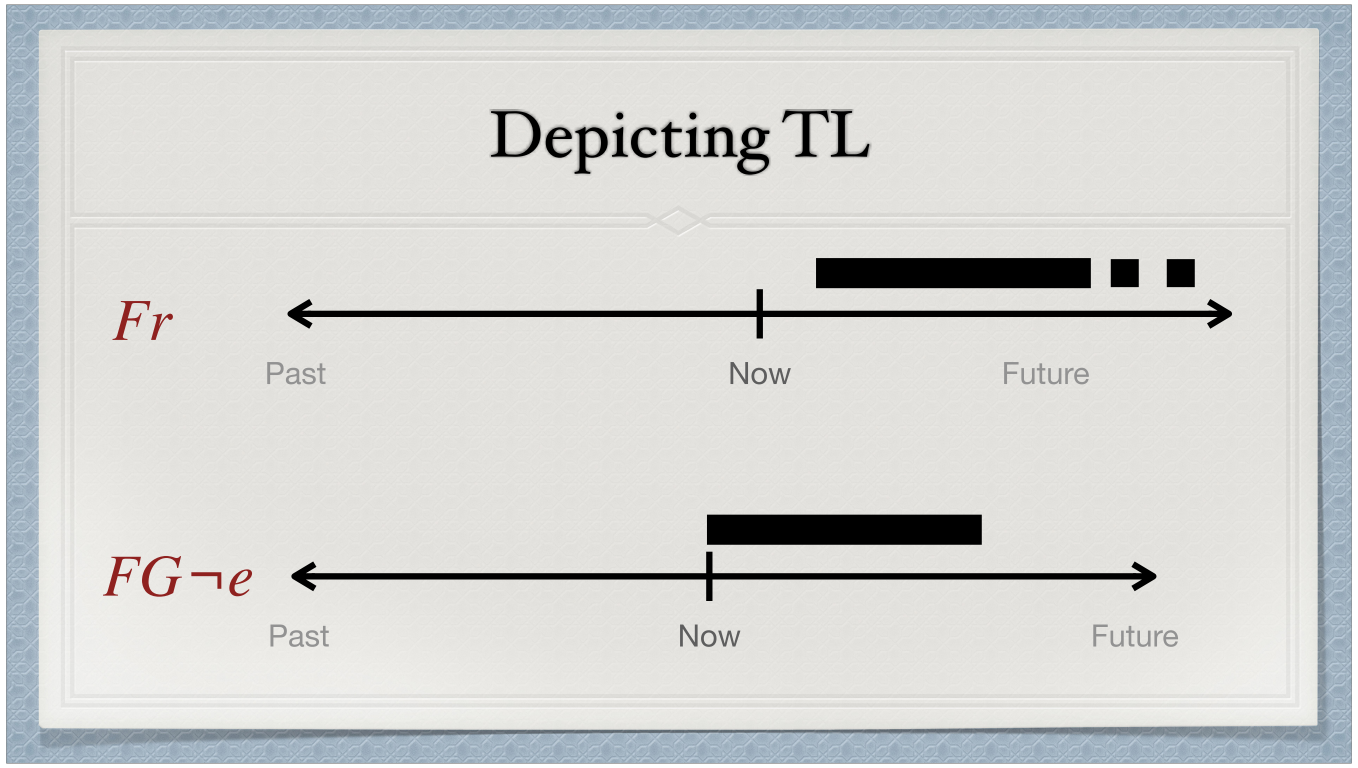
Religion dwindled throughout his life, possibly due to his work on freedom and time
No background in mathematics or math logic
Opposed purely formal logic and very much philosophical view: disliked idealism and preferred realism; logic useless unless it helps describe the world
ANP developed systems of tense logic based on ordering of events in time and operators to represent tensed speech;
series of John Locke lectures given in Oxford 1955--6 and published in a 1957 book Time and Modality
by 1967, enough work on tense logic had exploded across the world~\cite{sep-prior} for ANP to write a sequel Past, Present, and Future in 1967
Interestingly, he noted in that there could be an application for discrete-time TL in digital computers



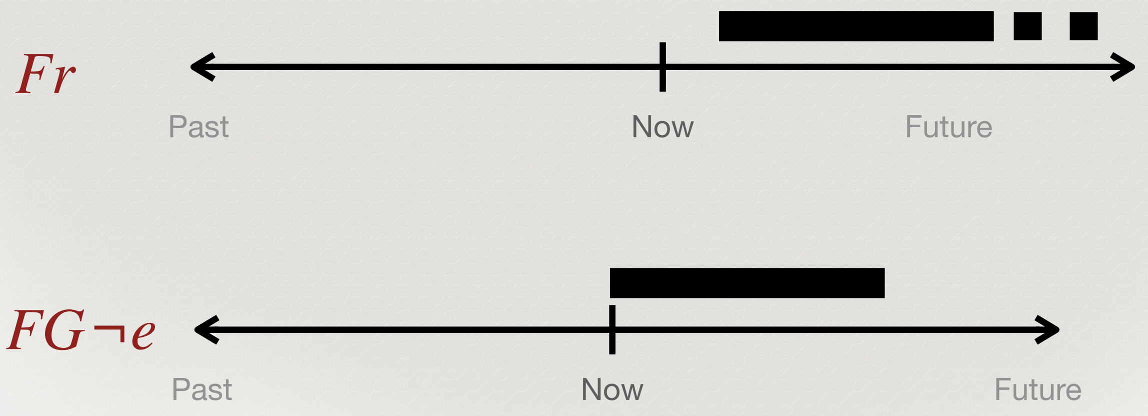
What is TL?

- Modal logic where possible worlds = instants
- Accessibility relation = events in sequence
- Basic operators F, G, X, P, H; computing uses modal ◊ □ ○
- Allows expression of statements about the timing of events
- Privileged *now* instant

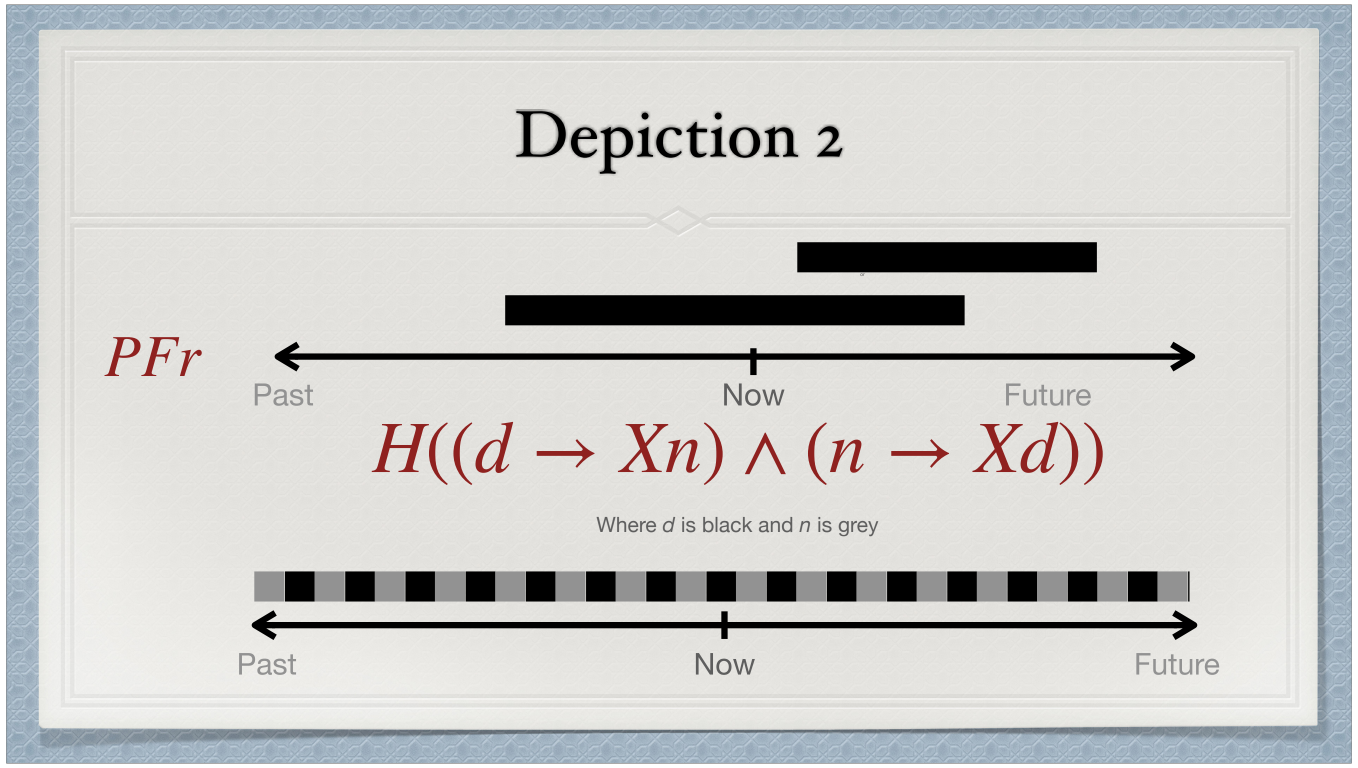
Propositional variables are Diodoran i.e. can be true at one time and false at another
Future, Global, Next, Past, Has always been
(Next operator not a ‘standard’ tense logic and is mentioned by Prior only in “non-standard logics” chapter where the idea is linked to Dana Scott)
Language contains a lot of tensed concepts and trying to translate them into logical system reveals the ambiguity!



Depicting TL



r: it rains—it's the classic example, although why something so renowned for being fickle as the weather is used. …
1: It will rain (at some point in the future for an unspecified duration)
2: Let e be "the Earth exists". Then top = at some point in the future, the Earth will be destroyed. Note the black line represents e: "in the future there will be an instant at which and forever after which e is not true"
More advanced in slideshow

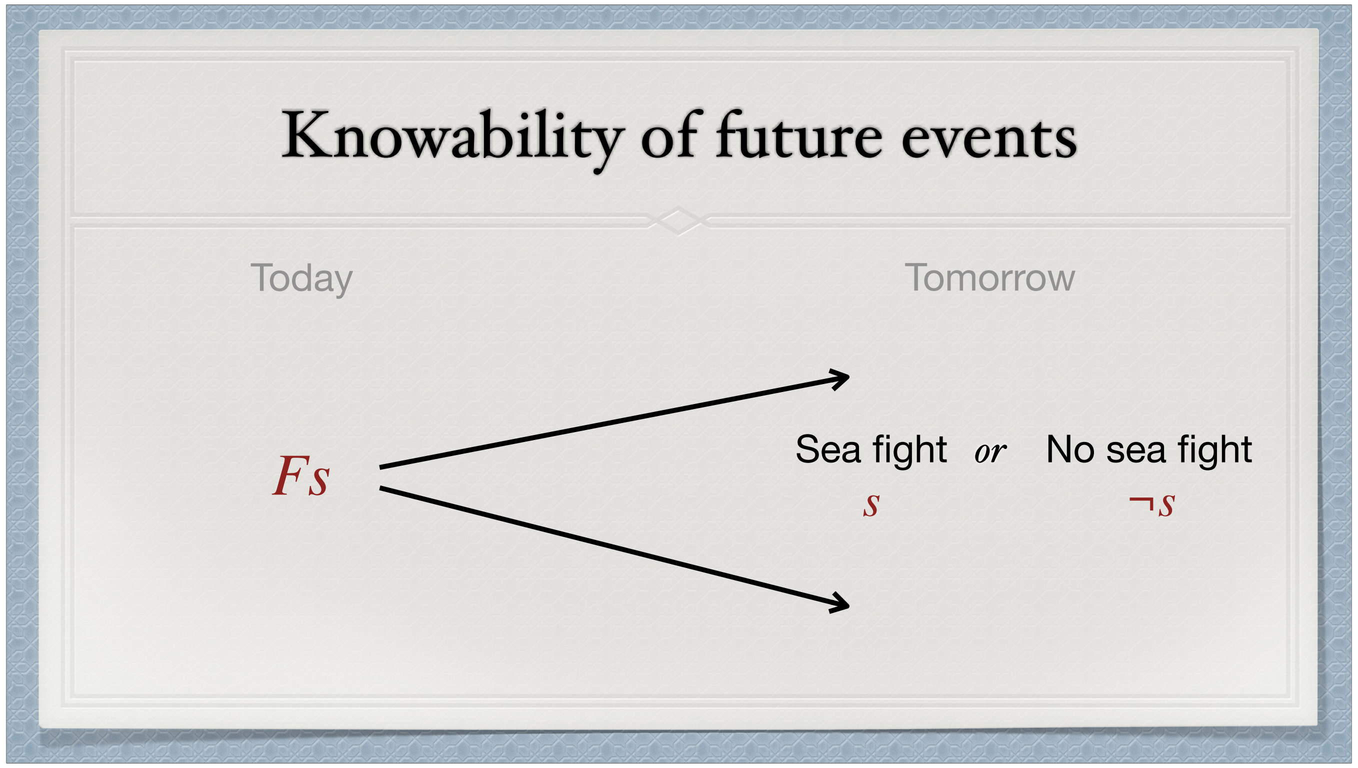


Depiction 2

$$H((d \rightarrow Xn) \wedge (n \rightarrow Xd))$$

Where *d* is black and *n* is grey

It looked like it was going to rain (a couple of possibilities here—note that a case in which it does not rain at all is basically the only incorrect case)
d = day, n = night; “It has always been the case that day follows night and vice versa”

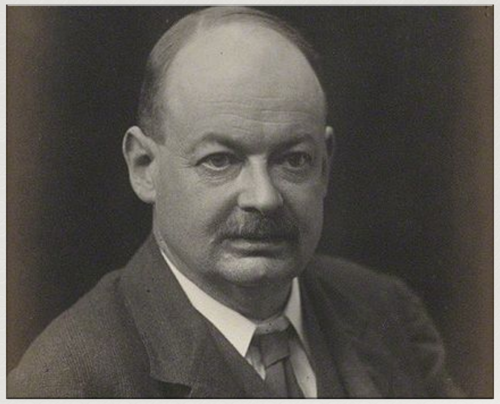


Knowability of future events

Aristotle's example. Future is inherently unknowable; what value does Fs have today?
Clearly one of them will be true so Fs must have a value. Is it *uncertain* i.e. there should be a trivalent logic? Or are all events are true until proven otherwise? Or perhaps Fs is true only if s is true in all possible futures? *click* this suggests a branching nature of future time, but linear past (since everything is known)
To many religions logicians, connected to determinism vs. Free will and the omniscience of God

Fundamental basis for time

- John McTaggart Ellis McTaggart (1866–1925)
- A-series: privileged now, operators indicate past and future
- B-series: logic of ordered instants
- McTaggart's bold claim



It's a good name (also his mother was called Ellen Ellis)
A continuous, operator first; B discrete, model first, operators use ordering of instants
Typically A-series seen as more 'logic' like and B-series as more 'mathematical'
McTaggart argued: B-theory cannot properly capture change; A-theory is inconsistent because present becomes past and changes its properties by so doing… so time is unreal
ANP worked seriously with both these series and preferred A
Ohrstrom and Hasle show how they are inter-translatable

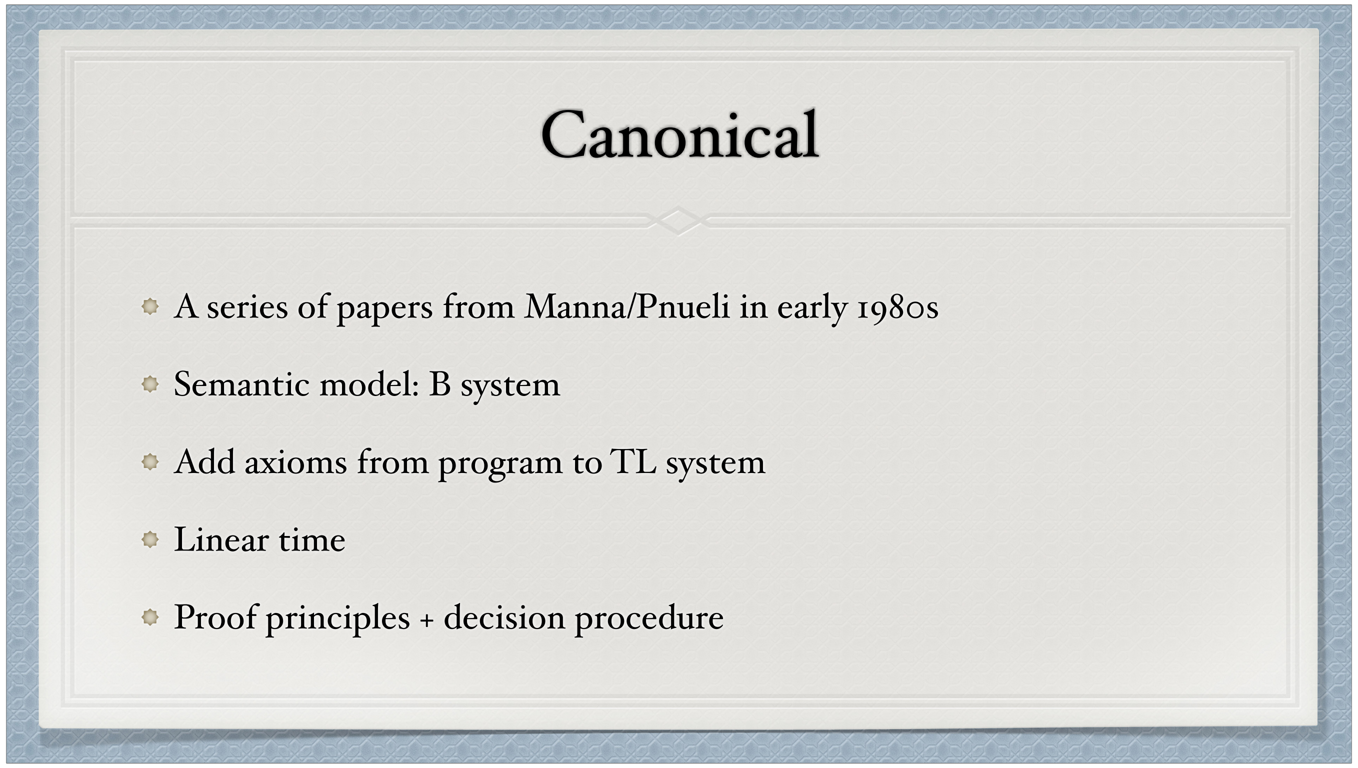
Glimpses in computing

In 1969 McCarthy and Hayes wrote a review paper explaining ways in which ideas from philosophy could be exploited for AI research; they talk about state-based representations and thoughts of ANP's tense logic for time and causality but the idea was not picked up
Manna working in late 1960s on ways to prove termination (inherently future) and building a way to do it based on translating program to a big old predicate and check its satisfiability
RMB in 1974 took the idea and uses an 'eventually' construct to prove termination: the idea is progress is captured in assertions which are true at least one time that control passes through them (rather than every time, as previous assertion methods like Floyd's used)
Notably RMB's method is much less technical than ZM's and also that he makes a link at the end with modal logic (citing R&U)
ZM developed idea further with Waldinger to create 'intermittent assertions'

Amir Pnueli (1941–2009)

- Israeli working from Weizmann, Stanford, Tel Aviv
- Non-terminating and eventually concurrent programs
- Inspiration from R&U to use modal assertions, calling it 'temporal logic'
- 1996 Turing Award winner for this work

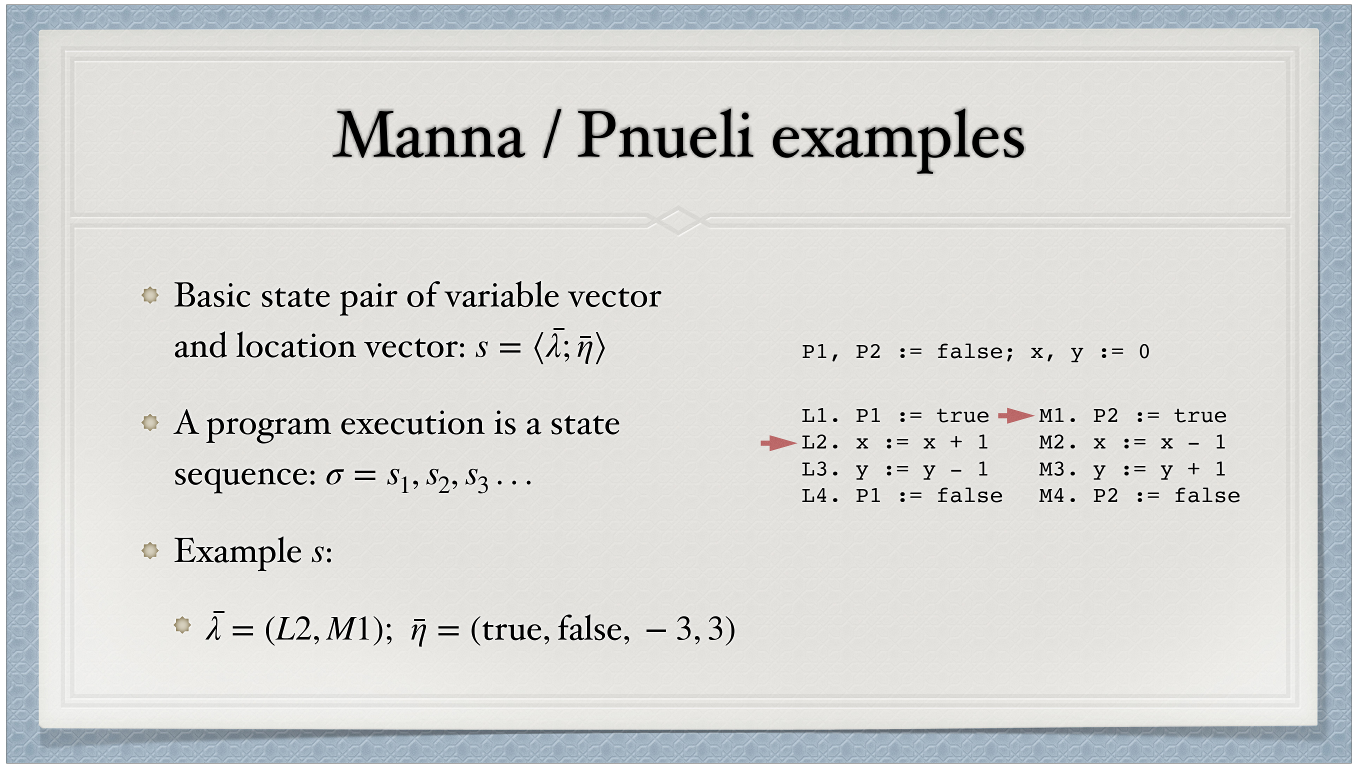
AP met ZM in Stanford and they worked together on ZM's satisfiability approach, trying it on various things
AP became interested in in cycling, (non-terminating) programs, using an explicit time variable in his predicates, moved to ZM's IAs
Back to Israel and founded computer science group in Tel Aviv, starting a research programme around TL
Then made the switch, with ZM, to using modal operators to manage concurrent programs in mid to late 1970s



Canonical

- A series of papers from Manna/Pnueli in early 1980s
- Semantic model: B system
- Add axioms from program to TL system
- Linear time
- Proof principles + decision procedure

View each instant in time as representing a program state; for the next instant, one process progresses
By treating processes as parts of the state, properties about the ordering of events can be stated with TL predicates
Useful for concurrency because relative ordering or simultaneity are important concepts for capturing concurrent properties—for example that no more than one process accesses a particular resource at a certain time
Little in the way of *application* to actual practical programs: for example, no real PL used (only algorithms); nothing of any real scale due to complexity
Not to say that ZMAP thought programming was unnecessary or beneath them; rather it was someone else's job to actually apply it



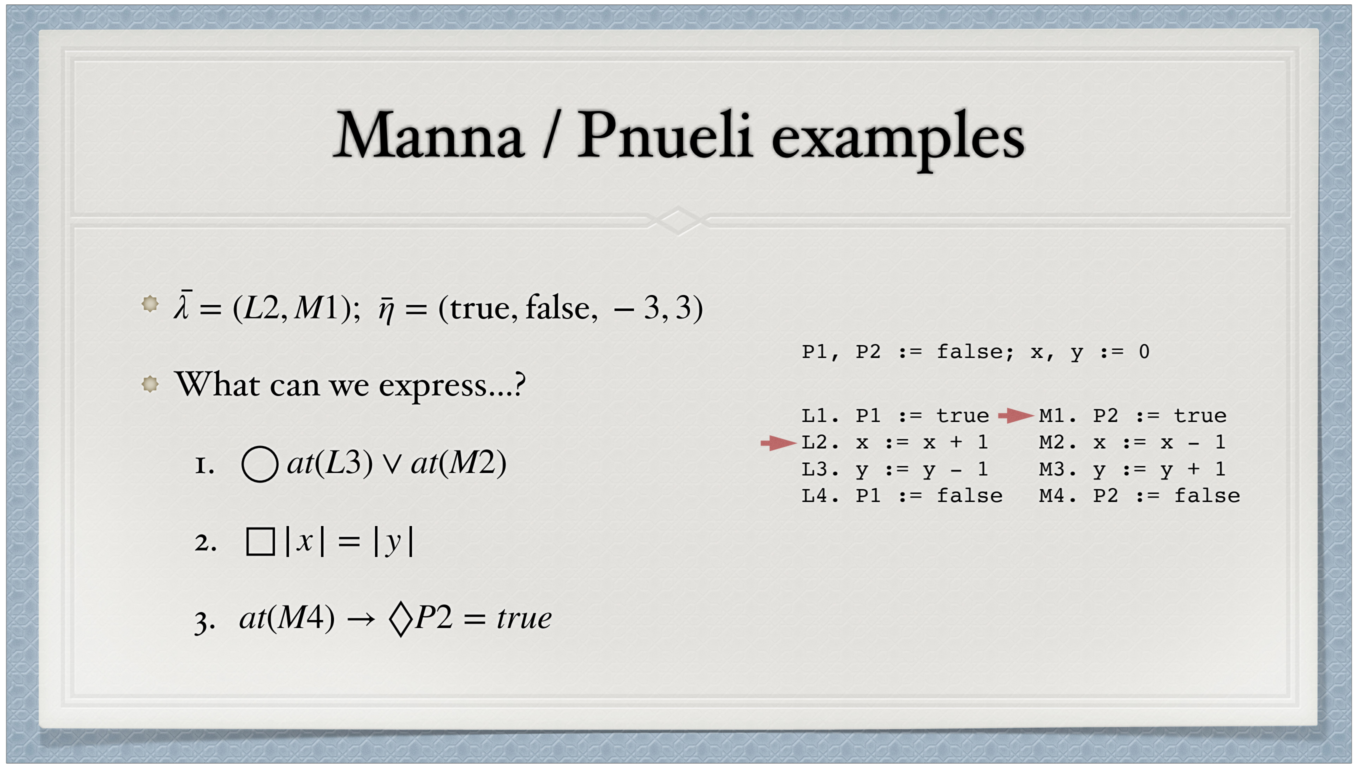
Manna / Pnueli examples

- Basic state pair of variable vector and location vector: $s = \langle \bar{\lambda}; \bar{\eta} \rangle$
- A program execution is a state sequence: $\sigma = s_1, s_2, s_3 \ldots$
- Example s:
    - $\bar{\lambda} = (L2, M1)$; $\bar{\eta} = (\text{true}, \text{false}, -3, 3)$

```
  P1, P2 := false; x, y := 0

  L1. P1 := true → M1. P2 := true
→ L2. x := x + 1   M2. x := x - 1
  L3. y := y - 1   M3. y := y + 1
  L4. P1 := false  M4. P2 := false
```

Lambda bar keeps track, for each process, which instruction comes next (or node in the transition graph); eta bar is values of variables
Click to show arrows



Manna / Pnueli examples

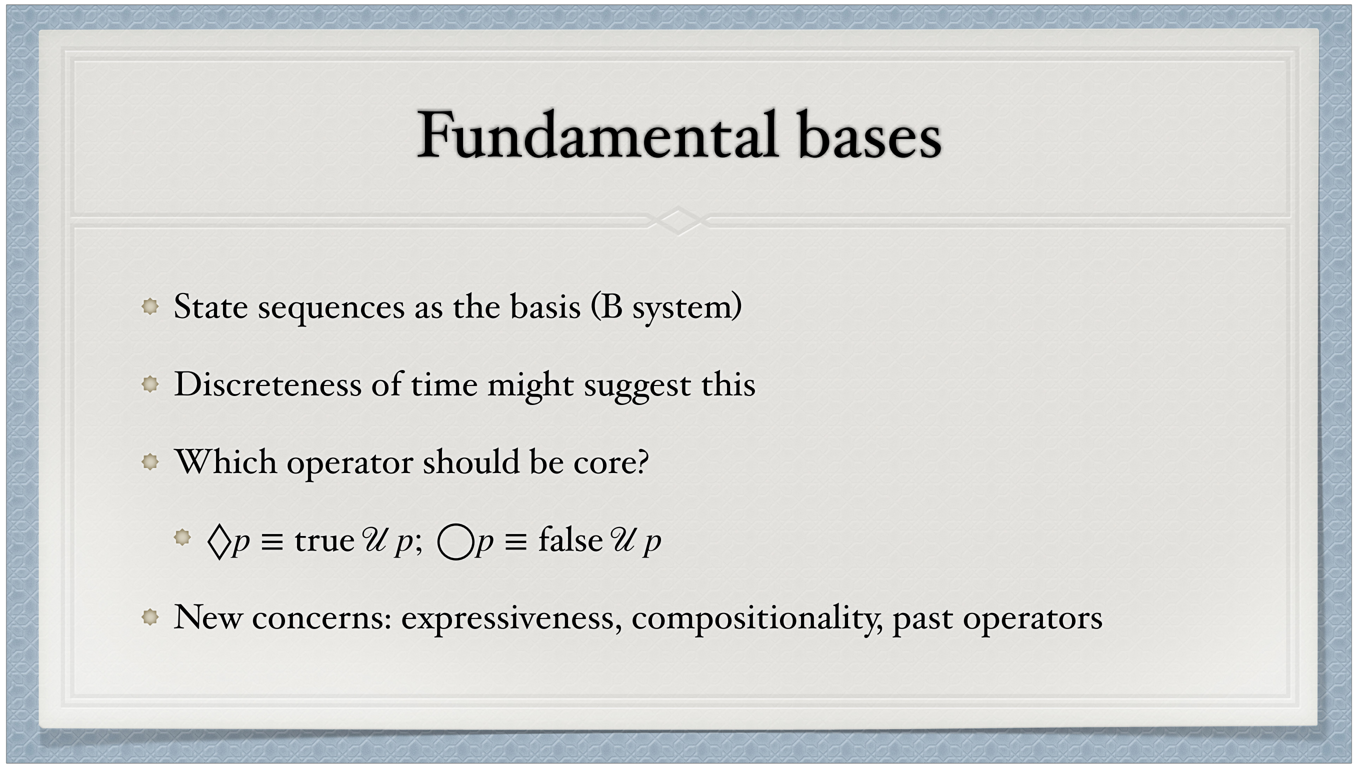
- $\bar{\lambda} = (L2, M1);\ \bar{\eta} = (\text{true}, \text{false}, -3, 3)$
- What can we express…?
  1. $\bigcirc\, at(L3) \vee at(M2)$
  2. $\Box\, |x| = |y|$
  3. $at(M4) \rightarrow \Diamond P2 = true$

```
  P1, P2 := false; x, y := 0

  L1. P1 := true →M1. P2 := true
→ L2. x := x + 1   M2. x := x - 1
  L3. y := y - 1   M3. y := y + 1
  L4. P1 := false  M4. P2 := false
```

Global view allows us to reason about multiple processes in a single assertion and to mix statements about the progress of processes with the values of variables, effectively indicating times

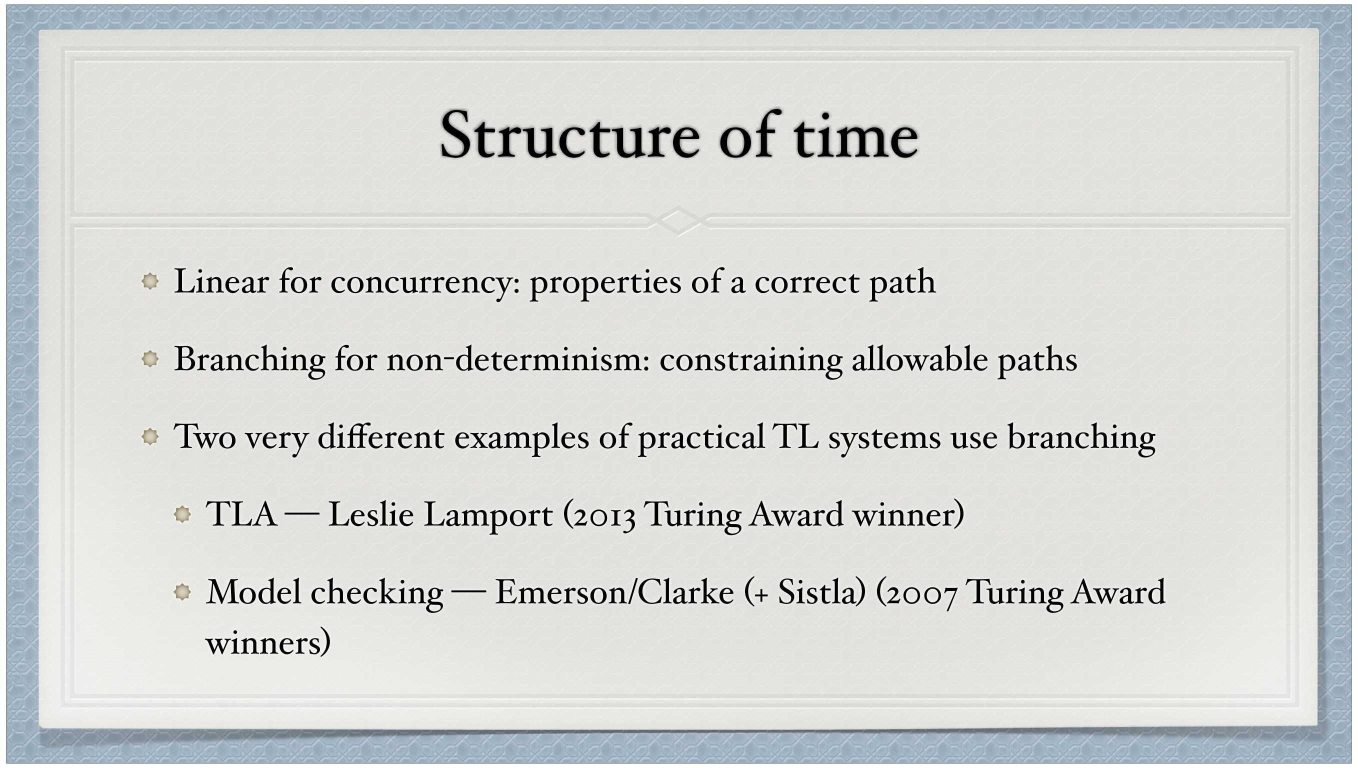
1. Yes (click)
2. No: what about if process 1 progresses next?
3. Depends if time is infinite, and system is fair… but probably (click)



Fundamental bases

- State sequences as the basis (B system)
- Discreteness of time might suggest this
- Which operator should be core?
  - $\Diamond p \equiv \text{true}\ \mathcal{U}\ p$; $\bigcirc p \equiv \text{false}\ \mathcal{U}\ p$
- New concerns: expressiveness, compositionality, past operators

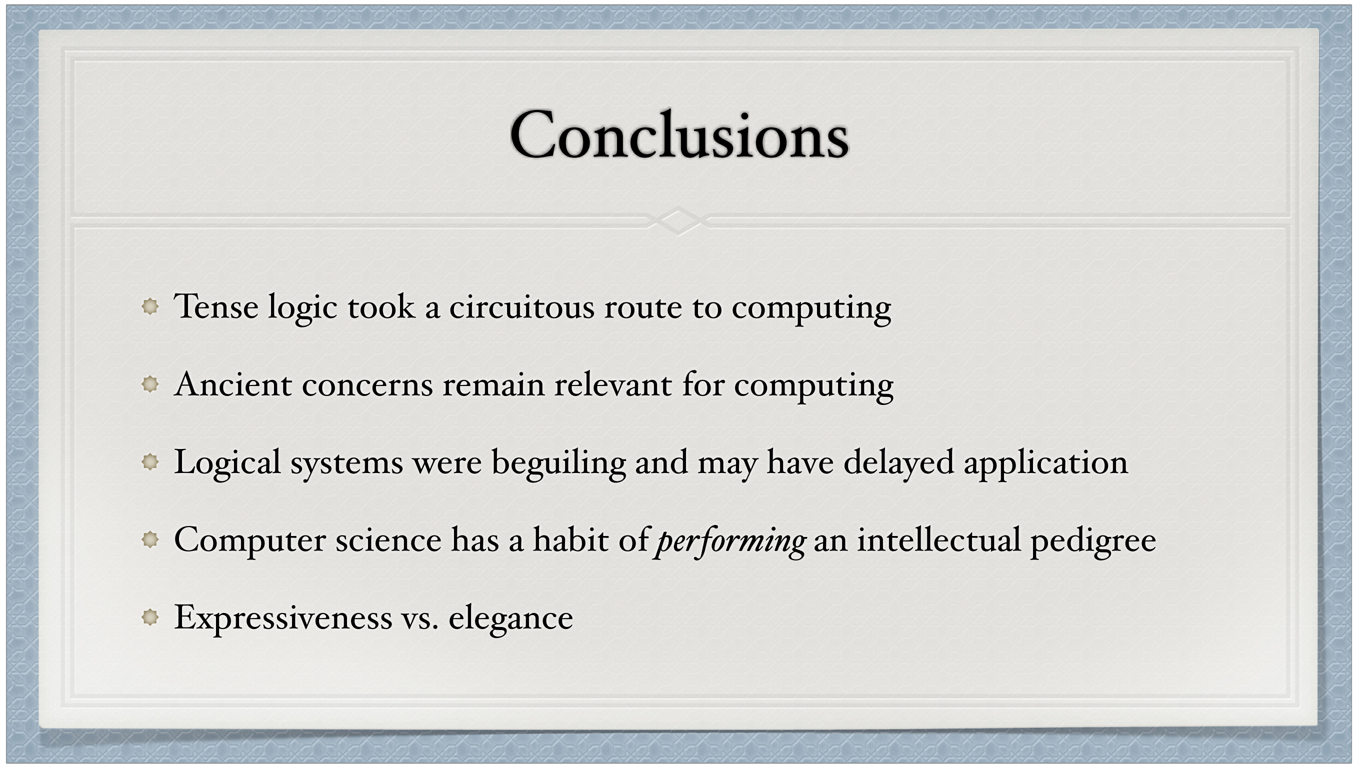
Ostensible focus on programming sets state sequences as the domain of discourse
ANP had already identified discrete time systems as potentially useful in digital computers
No exploration of an operator-first approach—why? Previous work on formalising computing started with “syntax–semantics” of PLs and later similar approach to programs; simply put, to be taken seriously as a bit of TCS, there must be a potential link with the practice of computing/programming, even if this is rarely exercised
Core operator: traditionally G; Gabbay, Pnueli, Shelah, Stavi show that U can do it all—based on a B system model
Time period where computers were being reframed as logic machines (see Priestley: computers weren’t actually built out of logic) and so why not play around in logic systems?
Less concern being paid to using the TL for programming tasks—certainly no examples being given—but this is somewhat par for the course in FM



Structure of time

- Linear for concurrency: properties of a correct path
- Branching for non-determinism: constraining allowable paths
- Two very different examples of practical TL systems use branching
  - TLA — Leslie Lamport (2013 Turing Award winner)
  - Model checking — Emerson/Clarke (+ Sistla) (2007 Turing Award winners)

Difference in expressiveness and operators have slightly different meanings!
Need to quantify over paths in branching time
TLA (temporal logic of actions) is a specification language which emphasises programming decisions and proposes using as little TL as possible, and keeping that very minimal
Emerson/Clarke's model checking: copes with scale problem by using "code skeletons"; translate these to a kind of branching logic and then run a decision algorithm over that to verify conditions



Conclusions

- Tense logic took a circuitous route to computing
- Ancient concerns remain relevant for computing
- Logical systems were beguiling and may have delayed application
- Computer science has a habit of *performing* an intellectual pedigree
- Expressiveness vs. elegance

ANP recognised applications for computers in 1967 but not properly applied until early 1970s
Time as branching vs. linear, moving from being a question of the knowability of the future or the potency of God; to one of concurrency vs. non-determinism
Penetration of ideas towards applications can take a long time; but the theory vs. practice spectrum is alive and well in computing and TL sits squarely in the former for quite a long time
Little real acknowledgement of long history of concern over these questions—instead reinventing the wheel (see LL's somewhat turgid relativity discussion)
An old dichotomy or spectrum raises its head once more: does adding extra operators improve effectiveness? Or is the increased complexity a problem? It depends on if you're interested in proving *about* or proving *with* …
TL provides an example of not exactly a new branch of logic per se or indeed something with obvious connections to practical programming—but rather as a new kind of boundary object, (often called TCS) which forms around old ideas put into new contexts

Selected references 1

- Mark Priestley. *A Science of Operations: Machines, Logic and the Invention of Programming.* History of Computing. Springer-Verlag, London, 2011.
- J M Ellis McTaggart. The unreality of time. *Mind*, 17(68):457–474, Oct 1908.
- A. N. Prior. *Time and Modality*. Oxford University Press, 1957 and *Present and Future*. Oxford University Press, 1967.
- Peter Øhrstrøm and Per Hasle. *Temporal Logic: from Ancient Ideas to Artificial Intelligence*, volume 57 of Studies in Linguistics and Philosophy. Springer Science & Business Media,
- N. Rescher and A. Urquart. *Temporal Logic*. Springer-Verlag, New York, 1971.
- B. Jack Copeland. Arthur Prior. *The Stanford Encyclopedia of Philosophy*. Stanford University, Spring 2020.

Selected references 2

- Z. Manna. *Termination of Algorithms.* PhD thesis, Carnegie-Mellon University, April 1968.
- J. McCarthy and P. J. Hayes. ‘Some philosophical problems from the standpoint of artificial intelligence’. In *Machine Intelligence* 4, pages 463–502. Edinburgh University Press, 1969.
- R. M. Burstall. ‘Program proving as hand simulation with a little induction’. In J. L. Rosenfeld, editor, *Information Processing, Proceedings of the 6th IFIP Congress* pages 308–312. North-Holland, 1974.
- Zohar Manna and Richard Waldinger. ‘Is “sometime” sometimes better than “always”?: intermittent assertions in proving program correctness’. Memo AIM-281, STAN-CS-76-558, Stanford Artificial Intelligence Laboratory, Jun 1976.
- N. Francez and A. Pnueli. ‘A proof method for cyclic programs’. *Acta Informatica*, 9:133–157, 1978.
- A. Pnueli. “The Temporal Semantics of Concurrent Programs”. In: *Theoretical Computer Science* 13.1 (1981), pp. 45–60.

Selected references 3

- Zohar Manna and Amir Pnueli. "Verification of Concurrent Programs: The Temporal Framework". In: *The Correctness Problem in Computer Science*. Ed. by R. S. Boyer and J. S. Moore. New York: Academic Press, May 1981, pp. 215–273.
- Z. Manna and A. Pnueli. "Verification of concurrent programs: a temporal proof system". In: *Proceedings of the 4th School of Advanced Programming*, Amsterdam, Holland. Amsterdam, Holland, June 1982.
- E Allen Emerson. "The beginning of model checking: A personal perspective". In: *25 Years of Model Checking*. Springer-Verlag, 2008, pp. 27–45.
- E Allen Emerson and Edmund M Clarke. "Characterizing cor- rectness properties of parallel programs using fixpoints". In: *International Colloquium on Automata, Languages, and Programming*. Springer-Verlag. 1980, pp. 169–181.
- Leslie Lamport. "The temporal logic of actions". In: *ACM Transactions on Programming Languages and Systems* 16.3 (May 1994), pp. 872–923.