

From Church and Prior to PSL Standing on The Shoulders of Giants

Moshe Y. Vardi

Rice University

Thread I: Classical Logic

Church, 1957: Use logic to specify sequential circuits.

Sequential circuits: $C = (I, O, R, f, g, R_0)$

- I : input signals
- O : output signals
- R : sequential elements
- $f : 2^I \times 2^R \rightarrow 2^R$: transition function
- $g : 2^R \rightarrow 2^O$: output function
- $R_0 \in 2^R$: initial assignment

Trace: element of $(2^I \times 2^R \times 2^O)^\omega$

$t = (I_0, R_0, O_0), (I_1, R_1, O_1), \dots$

- $R_{j+1} = f(I_j, R_j)$
- $O_j = g(R_j)$

Specifying Traces

View infinite trace $t = (I_0, R_0, O_0), (I_1, R_1, O_1), \dots$ as a mathematical structure:

- Domain: N
- Binary relations: $<, \leq$
- Unary relations: $I \cup R \cup O$

First-Order Logic (FO):

- Unary atomic formulas: $P(x)$ ($P \in I \cup R \cup O$)
- Binary atomic formulas: $x < y, x \leq y$

Example: $(\forall x)(\exists y)(x < y \wedge P(y))$ – P holds i.o.

Monadic Second-Order Logic (MSO):

- Monadic second-order quantifier: $\exists Q$
- New unary atomic formulas: $Q(x)$

Model-Checking Problem: Given circuit C and formula φ , does φ hold in all traces of C .

Easy Observation: Model-checking problem reducible to satisfiability problem – use FO to encode the “logic” (i.e., f, g) of the circuit C .

Büchi Automata

Büchi Automaton: $A = (\Sigma, S, S_0, \rho, F)$

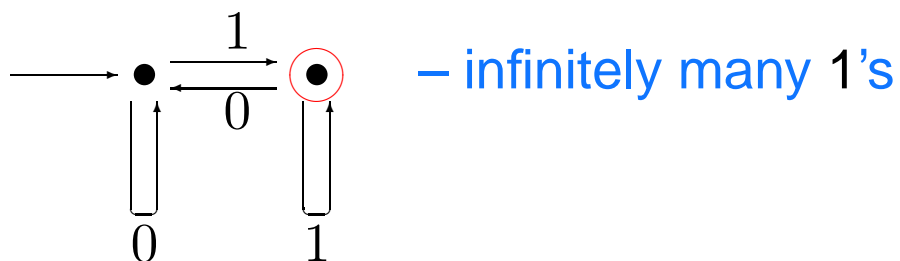
- *Alphabet:* Σ
- *States:* S
- *Initial states:* $S_0 \subseteq S$
- *Transition relation:* $\rho \subseteq S \times \Sigma \times S$
- *Accepting states:* $F \subseteq S$

Input word: a_0, a_1, \dots

Run: s_0, s_1, \dots

- $s_0 \in S_0$
- $(s_i, a_i, s_{i+1}) \in \rho$ for $i \geq 0$

Acceptance: F visited infinitely often



Logic vs. Automata

Paradigm: Compile high-level logical specifications into low-level finite-state language

Compilation-Theorem: [Büchi, 1960]

Given an MSO formula φ , one can construct a Büchi automaton A_φ such that a trace σ satisfies φ if and only if σ is accepted by A_φ .

MSO Satisfiability Algorithm:

1. φ is satisfiable iff $L(A_\varphi) \neq \emptyset$
2. $L(\Sigma, S, S_0, \rho, F) \neq \emptyset$ iff there is a path from S_0 to a state $f \in F$ and a cycle from f to itself.

Corollary [Church, 1960]: Model checking sequential circuits wrt MSO specs is decidable.

Complexity

Nonelementary Growth:

$$2^{\dots 2^n}$$

(tower of height $O(n)$)

Analysis of Büchi's Construction: nonelementary upper bound

Church, 1960: Algorithm not very efficient

Lower Bound [Stockmeyer, 1974]: Satisfiability of FO over finite words is nonelementary (extension to infinite words is easy).

Thread II: Temporal Logic

Prior, 1914–1969, Philosophical Preoccupations:

- *Religion*: Methodist, Presbyterian, atheist, agnostic
- *Ethics*: “Logic and The Basis of Ethics”, 1949
- *Free Will, Predestination, and Foreknowledge*:
 - “The future is to some extent, even if it is only a very small extent, something we can make for ourselves”.
 - “Of what will be, it has now been the case that it will be.”
 - “There is a deity who infallibly knows the entire future.”

Mary Prior: “I remember his waking me one night [in 1953], coming and sitting on my bed, . . . , and saying he thought one could make a formalised tense logic.”

- 1955–6: John Locke Lecturer at Oxford
- 1957: “Time and Modality”

Linear vs. Branching Time, A

- **Prior's** first lecture on tense logic, Wellington University, 1954: linear time.
- **Prior's** "Time and modality", 1957: relationship between tense logic and modal logic.
- Sep. 1958, letter from **Kripke**: "[I]n an indetermined system, we perhaps should not regard time as a linear series, as you have done. Given the present moment, there are several possibilities for what the next moment may be like – and for each possible next moment, there are several possibilities for the moment after that. Thus the situation takes the form, not of a linear sequence, but of a 'tree'". (**Kripke** was a high-school student, not quite 18, in Omaha, Nebraska.)
- **Kripke** became interested in modal logic through reading **Prior's** 'Modality and Quantification in S5' in 1956. His letter was in response to **Prior's** "Time and Modality".

Linear vs. Branching Time, B

- Prior developed the idea into Ockhamist and Peircean theories of branching time (branching-time logic *without* path quantifiers)

Sample formula: $CKM_pM_qAMK_pM_qMK_qMp$

- Burgess, 1978: “Prior would agree that the determinist sees time as a line and the indeterminist sees times as a system of forking paths.”

Linear vs. Branching Time, C

Philosophical Conundrum

- Prior:

- Nature of course of time – branching
- Nature of course of events – linear

- Rescher:

- Nature of time – linear
- Nature of course of events – branching
- “We have 'branching *in* time', not 'branching *of* time”.

Linear time: Bull, Cocchiarella, Kamp, Scott and others continued the development of linear time during the 1960s.

Temporal and Classical Logics

Key Theorems:

- **Kamp, 1968:** Linear temporal logic with past and binary temporal connectives (“until” and “since”) has precisely the expressive power of FO over the integers.
- **Thomas, 1979:** FO over naturals has the expressive power of star-free ω -regular expressions (MSO= ω -regular).

Precursors:

- **Büchi, Elgot, Trakhtenbrot, 1960:** On finite words, MSO=RE
- **McNaughton & Papert, 1971:** On finite words, FO=star-free-RE

The Temporal Logic of Programs

Precursors:

- **Prior**: “There are practical gains to be had from this study too, for example in the representation of time-delay in computer circuits”
- **Rescher & Urquhart, 1971**: applications to processes (“a programmed sequence of states, deterministic or stochastic”)

“Big Bang 1” [Pnueli, 1977]:

- Future linear temporal logic (LTL) as a logic for the specification of non-terminating programs
- Temporal logic with “next” and “until”
- Model checking via reduction to MSO

Expressive Power

Gabbay, Pnueli, Shelah & Stavi, 1980: Propositional LTL has precisely the expressive power of FO over the naturals.

$LTL=FO=star\text{-free } \omega\text{-RE} < MSO=\omega\text{-RE}$

Meyer on LTL, 1980, in “Ten Thousand and One Logics of Programming”:

“The corollary due to Meyer – I have to get in my controversial remark – is that that [GPSS’80] makes it theoretically uninteresting.”

Computational Complexity

Recall: Satisfiability of FO over traces is non-elementary

Contrast with LTL:

- Wolper, 1981: LTL satisfiability is in EXPTIME.
- Sistla & Clarke, 1982: LTL satisfiability and model checking is PSPACE-complete.

Basic Technique: *tableaux* (influenced by branching-time techniques)

Model Checking

“Big Bang 2” [Clarke & Emerson, Queille & Sifakis, 1981]: Model checking programs of size m wrt CTL formulas of size n can be done in time mn .

Linear-Time Response [Lichtenstein & Pnueli, 1985]: Model checking programs of size m wrt LTL formulas of size n can be done in time $m2^{O(n)}$ (tableau-based).

Back to Automata

Exponential-Compilation Theorem:

[V. & Wolper, 1983–1986]

Given an LTL formula φ of size n , one can construct a Büchi automaton A_φ of size $2^{O(n)}$ such that a trace σ satisfies φ if and only if σ is accepted by A_φ .

Automata-Theoretic Algorithms:

1. *LTL Satisfiability*: φ is satisfiable iff $L(A_\varphi) \neq \emptyset$
(PSPACE)
2. *LTL Model Checking*: $M \models \varphi$ iff $L(M \times A_{\neg\varphi}) = \emptyset$
($m2^{O(n)}$)

Reduction to Practice

Practical Theory:

- Courcoubetis, V., Yannakakis & Wolper, 1989: Optimized search algorithm for explicit model checking
- Burch, Clarke, McMillan, Dill & Hwang, 1990: Symbolic algorithm for LTL compilation
- Clarke, Grumberg & Hamaguchi, 1994: Optimized symbolic algorithm for LTL compilation
- Gerth, Peled, V. & Wolper, 1995: Optimized explicit algorithm for LTL compilation

Implementation:

- COSPAN [Kurshan, 1983]: deterministic automata specs
- Spin [Holzmann, 1995]: Promela w. LTL:
- SMV [McMillan, 1995]: SMV w. LTL

Satisfactory solution to Church's problem? Almost, but not quite, since $LTL < MSO = \omega\text{-RE}$.

Enhancing Expressiveness

- Wolper, 1981: Enhance LTL with grammar operator, retaining EXPTIME-ness (PSPACE [SC'82])
- V. & Wolper, 1983: Enhance LTL with automata, retaining PSPACE-completeness
- Sistla, V. & Wolper, 1985: Enhance LTL with quantification, losing elementariness
- V., 1989: Enhance LTL with fixpoints, retaining PSPACE-completeness

Bottom Line: ETL (LTL w. automata) = μ TL (LTL w. fixpoints) = MSO, and has exponential-compilation property.

Independently Kurshan, 1983–: Use ω -automata for specification and modeling of protocols; development of COSPAN

Thread III: Dynamic and Branching-Time Logics

Dynamic Logic [Pratt, 1976]:

- The $\Box\varphi$ of modal logic can be taken to mean “ φ holds after an execution of a program step”.
- Dynamic modalities:
 - $\langle\alpha\rangle\varphi$ – φ holds after some execution of α ,
 - $[\alpha]\varphi$ – φ holds after all executions of α .
 - $\psi \rightarrow [\alpha]\varphi$ corresponds to Hoare triple $\{\psi\}\alpha\{\varphi\}$.

Propositional Dynamic Logic [Fischer & Ladner, 1977]: *Boolean* propositions, programs – regular expressions over *atomic* programs.

Satisfiability [Pratt, 1978]: EXPTIME – using tableau-based algorithm

Extensions to nonterminating programs [Streett 1981, Harel & Sherman 1981] – awkward compared to linear temporal logic.

Branching-Time Logic

From dynamic logic back to temporal logic: The dynamic-logic view is clearly branching; what is the analog for temporal logic?

- Emerson & Clarke, 1980: correctness properties as fixpoints over computation trees
- Ben-Ari, Manna & Pnueli, 1981: branching-time logic UB; satisfiability in EXPTIME using tableaux
- Clarke & Emerson, 1981: branching-time logic CTL; efficient model checking
- Emerson & Halpern, 1983: branching-time logic CTL* – ultimate branching-time logic

Key Idea: Prior missed *path quantifiers*

Combining Dynamic and Temporal Logics

Two distinct perspectives:

- Temporal logic: *state based*
- Dynamic logic: *action based*

Symbiosis:

- Harel, Kozen & Parikh, 1980: Process Logic (branching time)
- V. & Wolper, 1983: Yet Another Process Logic (branching time)
- Harel and Peleg, 1985: Regular Process Logic (linear time)
- Henriksen and Thiagarajan, 1997: Dynamic LTL (linear time)
- Beer, Ben-David & Landver, 1998: RCTL (branching time)
- Beer, Ben-David, Eisner, Fisman, Gringauze, Rodeh, 2001: Sugar

Distinction: RCTL and Sugar are state based (no actions)

Thread IV: From ForSpec to PSL

Model Checking at Intel

Prehistory:

- 1990: feasibility study using COSPAN (precursor of FormalCheck)
- 1992: a pilot project using SMV
- 1995: an internally developed (linear time) property-specification language

History:

- 1997: Development of 2nd-generation technology started (engine and language)
- 1999: BDD-based model checker released
- 2000: SAT-based model checker released
- 2000: ForSpec (language) released

Dr. Vardi Goes to Intel

1997: (w. Fix, Hadash & Sananes)

V.: How about LTL?

F., H. & S.: Not expressive enough.

V.: How about ETL?

F., H. & S.: Users will object.

1998 (w. Landver)

V.: How about ETL?

L.: Users will object.

L.: How about regular expressions?

V.: They are equivalent to automata.

RELTL: LTL plus dynamic modalities, interpreted linearly – $\langle e \rangle \varphi$, $[e] \varphi$

Easy: RELTL=ETL= ω -RE

ForSpec: RELTL + hardware features (clocks and resets) [Armoni, Fix, Flaisher, Gerth, Ginsburg, Kanza, Landver, Mador-Haim, Singerman, Tiemeyer, V., Zbar]

From ForSpec to PSL

Standardization:

- Process started in 2000
- Four candidates: IBM's Sugar, Intel's ForSpec, Mororola's CBV, and Verisity's E.
- Fierce debate on linear vs. branching time

Outcome:

- Big political win for IBM (see references to PSL/Sugar)
- Big technical win for Intel
 - PSL is LTL + RE + clocks + resets
 - Branching-time extension as an acknowledgement to Sugar
 - Some evolution over time in hardware features
- Major influence on the design of SVA.

Some Philosophical Points

- Science is a cathedral; we are the masons.
- There is no architect; outcome is unpredictable.
- Most of our contributions are smaller than we'd like to think.
- Even small contributions can have major impact.
- Much is forgotten and has to be rediscovered.