# From Löwenheim to Pnueli, from Pnueli to PSL and SVA

Moshe Y. Vardi

Rice University

# Thread I: Monadic Logic

**Monadic Class**: First-order logic with $=$ and monadic predicates – captures *syllogisms*.

- $(\forall x)P(x), (\forall x)(P(x) \rightarrow Q(x)) \models (\forall x)Q(x)$

> [Löwenheim, 1915]: The Monadic Class is decidable.
> - *Proof*: Bounded-model property – if a sentence is satisfiable, it is satisfiable in a structure of bounded size.
> - *Proof technique*: quantifier elimination.

**Monadic Second-Order Logic**: Allow second-order quantification on monadic predicates.

> [Skolem, 1919]: Monadic Second-Order Logic is decidable – via bounded-model property and quantifier elimination.

**Question**: What about $<$?

# Thread II: Sequential Circuits

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 $(2^I \times 2^R \times 2^O)^\omega$

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

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

# Specifying Traces

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

- Domain: *N*
- Binary relation: $<$
- 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: $x < y$

**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 $\varphi$; does $\varphi$ 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**: $A = (\Sigma, S, S_0, \rho, F)$

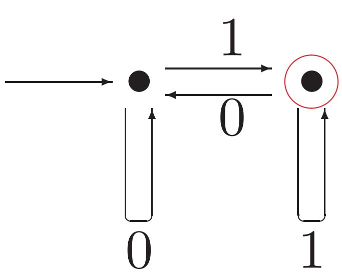
- *Alphabet*: $\Sigma$
- *States*: $S$
- *Initial states*: $S_0 \subseteq S$
- *Transition function*: $\rho : S \times \Sigma \rightarrow 2^S$
- *Accepting states*: $F \subseteq S$

**Input word**: $a_0, a_1, \ldots$

**Run**: $s_0, s_1, \ldots$

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

**Acceptance**: $F$ visited infinitely often




– infinitely many 1's

**Fact**: Büchi automata define the class $\omega$-*Reg* of $\omega$-regular languages.

# Logic vs. Automata

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

**Compilation-Theorem**: [Büchi,1960] Given an MSO formula $\varphi$, one can construct a Büchi automaton $A_\varphi$ such that a trace $\sigma$ satisfies $\varphi$ if and only if $\sigma$ is accepted by $A_\varphi$.

**MSO Satisfiability Algorithm**:

1. $\varphi$ 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.

Church, 1960: "Algorithm not very efficient" (*nonelementary complexity*, [Stockmeyer, 1974]).

# Thread III: Temporal Logic

Prior, 1914–1969, Philosophical Preoccupations:

- *Religion*: Methodist, Presbytarian, 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.”

- 1957: “Time and Modality”

# Temporal and Classical Logics

**Key Theorem**:

- Kamp, 1968: Linear temporal logic with past and binary temporal connectives (“until” and “since”) has precisely the expressive power of FO over the integers.

# 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 "always"and "eventually" (later, "next" and "until")
- Model checking via reduction to MSO and automata

**Crux**: Need to specify ongoing behavior rather than input/output relation!

# Linear Temporal Logic

**Linear Temporal logic** (LTL): logic of temporal sequences (Pnueli, 1977)

*Main feature*: time is implicit

- *next* $\varphi$: $\varphi$ holds in the next state.
- *eventually* $\varphi$: $\varphi$ holds eventually
- *always* $\varphi$: $\varphi$ holds from now on
- $\varphi$ *until* $\psi$: $\varphi$ holds until $\psi$ holds.

- $\pi, w \models next\ \varphi$ if $w$ $\bullet \longrightarrow \underset{\varphi}{\bullet} \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \ldots$

- $\pi, w \models \varphi\ until\ \psi$ if $w$ $\underset{\varphi}{\bullet} \longrightarrow \underset{\varphi}{\bullet} \longrightarrow \underset{\varphi}{\bullet} \longrightarrow \underset{\psi}{\bullet} \longrightarrow \bullet \ldots$

# Examples

- always not ($CS_1$ and $CS_2$): mutual exclusion (safety)
- always (Request implies eventually Grant): liveness
- always (Request implies (Request until Grant)): liveness
- always (always eventually Request) implies eventually Grant: liveness

# Expressive Power

- Gabbay, Pnueli, Shelah & Stavi, 1980: Propositional LTL has precisely the expressive power of FO over the naturals.
- Thomas, 1979: FO over naturals has the expressive power of star-free $\omega$-regular expressions
- LTL=FO=star-free $\omega$-RE $<$ MSO=$\omega$-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.
- Halpern & Reif, 1981, Sistla & Clarke, 1982: LTL satisfiability is PSPACE-complete.

**Basic Technique**: *tableau*

# Model Checking

**"Big Bang 2"** [Clarke & Emerson, 1981, Queille & Sifakis, 1982]: Model checking programs of size $m$ wrt CTL formulas of size $n$ can be done in time $mn$.

**Note**: CTL was a slight extension of UB, a branching-time logic introduce in [Ben-Ari, Manna, Pnueli, 1981].

**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).

**Seemingly**:

- *Automata*: Nonelementary
- Tableaux: exponential

# Back to Automata

**Exponential-Compilation Theorem**:

[V. & Wolper,1983–1986]

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

**Automata-Theoretic Algorithms**:

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

# Enhancing Expressiveness

- Wolper, 1981: Enhance LTL with grammar operators, retaining EXPTIME-ness (PSPACE [SC'82])
- V. & Wolper, 1983: Enhance LTL with automata, retaining PSPACE-completeness
- Sistla, V. & Wolper, 1985: Enhance LTL with 2nd-order quantification, losing elementariness
- V., 1989: Enhance LTL with fixpoints, retaining PSPACE-completeness

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

# Thread IV: From Philosophy to Industry

**Dr. Vardi Goes to Intel**:

1997: (w. Fix, Hadash, Kesten, & Sananes)

V.: How about LTL?
F., H., K., & S.: Not expressive enough.

V.: How about ETL? $\mu$TL?
F., H., K., & 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-logic modalities, interpreted linearly – $[e]\varphi$

**Easy**: RELTL=$\omega$-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 and SVA

**Industrial Standardization**:
- Process started in 2000
- Four candidates: IBM's Sugar, Intel's ForSpec, Mororola's CBV, and Verisity's E.

**Outcome**:

- Big political win for IBM (see references to PSL/Sugar)
- Big technical win for Intel
  - PSL is essentially LTL + RE + clocks + resets
  - Some evolution over time in hardware features
- Major influence on the design of *SVA* (another industrial standard)

**Bottom Line**: *Huge* push for model checking in industry.

# Pnueli's Seminal Contributions

- Applying an obscure philosophical logic (LTL) to computer-science problems
  - Reasoning about ongoing behavior
  - Ease of use
  - Computational tractability
- Facilitating the emergence of model checking by introducing branching-time logic
- Showing that LTL has an exponential-time model-checking algorithm