

A Game-Theoretic Approach to Automated Program Generation

Moshe Y. Vardi

Rice University

Model Checking

Model Checking:

- *Given:* Program P , Specification φ .
- *Task:* Check that $P \models \varphi$

Success:

- *Algorithmic methods:* temporal specifications and finite-state programs.
- *Also:* Certain classes of infinite-state programs
- *Tools:* SMV, SPIN, SLAM, etc.
- *Impact:* on industrial design practices is increasing.

Problems:

- Designing P is hard and expensive.
- Redesigning P when $P \not\models \varphi$ is hard and expensive.

Synthesis=Automated Design

Basic Idea:

- Start from spec φ , design P such that $P \models \varphi$.

Advantage:

- No verification
- No re-design

- Derive P from φ algorithmically.

Advantage:

- No design

In essence: Declarative programming taken to the limit.

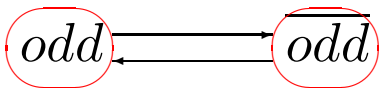
Synthesis of Ongoing Programs

Specs: Temporal logic formulas

Early 1980s: Satisfiability approach
(Wolper, Clarke+Emerson, 1981)

- *Given:* φ
- *Satisfiability:* Construct $M \models \varphi$
- *Synthesis:* Extract P from M .

Example: $always(odd \rightarrow next \neg odd) \wedge$
 $always(\neg odd \rightarrow next odd)$



Reactive Systems

Reactivity: Interaction with environment
(Harel+Pnueli, 1985)

Example: Printer specification –

J_i - job i submitted, P_i - job i printed.

- **Safety:** two jobs are not printed together
always $\neg(P_1 \wedge P_2)$
- **Liveness:** every jobs is eventually printed
always $\bigwedge_{j=1}^2 (J_j \rightarrow \text{eventually } P_j)$

Satisfiability and Synthesis

Specification Satisfiable? Yes!

Model M: A single state where J_1 , J_2 , P_1 , and P_2 are all false.

Extract program from M ? No!

Why? Because M handles only one input sequence.

- J_1, J_2 : input variables, controlled by environment
- P_1, P_2 : output variables, controlled by system

Desired: a system that handles *all* input sequences.

Conclusion: Satisfiability is inadequate for synthesis.

Realizability

I : input variables

O : output variables

Game:

- *System*: choose from 2^O
- *Env*: choose from 2^I

Infinite Play:

i_0, i_1, i_2, \dots

o_0, o_1, o_2, \dots

Infinite Behavior: $i_0 \cup o_0, i_1 \cup o_1, i_2 \cup o_2, \dots$

Win: behavior \models spec

Specifications: LTL formula on $I \cup O$

Strategy: Function $f : (2^I)^* \rightarrow 2^O$

Realizability: Pnueli+Rosner, 1989

Existence of winning strategy for specification.

Church's Problem

Church, 1963: Realizability problem wrt specification expressed in S1S (monadic second-order theory of one successor function)

Büchi+Landweber, 1969:

- Realizability is decidable.
- If a winning strategy exists, then a finite-state winning strategy exists.
- Realizability algorithm produces finite-state strategy.

Rabin, 1972: Simpler solution via Rabin tree automata.

Question: LTL is subsumed by S1S, so what did Pnueli and Rosner do?

Answer: better algorithms!

Strategy Trees

Infinite Tree: D^* (D - directions)

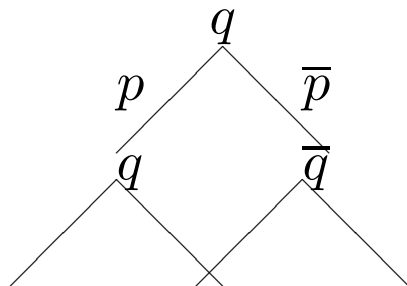
- **Root:** ε
- **Children:** $xd, x \in D^*, d \in D$

Labeled Infinite Tree: $\tau : D^* \rightarrow \Sigma$

Strategy: $f : (2^I)^* \rightarrow 2^O$

Rabin's Insight: A strategy is a labeled tree with directions $D = 2^I$ and alphabet $\Sigma = 2^O$.

Example: $I = \{p\}, O = \{q\}$



Winning: Every branch satisfies spec.

Automata on Infinite k -ary Trees

$$A = (\Sigma, S, S_0, \rho, \alpha)$$

- Σ : finite alphabet
- S : finite state set
- $S_0 \subseteq S$: initial state set
- ρ : transition function
 - $\rho : S \times \Sigma \rightarrow 2^{S^k}$
- α : acceptance condition
 - $\alpha = \{(G_1, B_1), \dots, (G_l, B_l)\}, G_i, B_i \subseteq S$
 - **Acceptance:** For some $(G_i, B_i) \in \alpha$, G_i is visited infinitely often, and B_i is visited finitely often.

Emptiness of Tree Automata

Emptiness: $L(A) = \emptyset$

Emptiness of Automata on Finite Trees: PTIME
test (Doner, 1965)

Emptiness of Automata on Infinite Trees: Difficult

- Rabin, 1969: non-elementary
- Hossley+Rackoff, 1972: 2EXPTIME
- Rabin, 1972: EXPTIME
- Emerson, Vardi+Stockmeyer, 1985: In NP
- Emerson+Jutla, 1991: NP-complete

Rabin's Realizability Algorithm

REAL(φ):

- Construct Rabin automaton A_φ that accepts all winning strategy trees for spec φ .
- Check non-emptiness of A_φ .
- If nonempty, then we have realizability; extract strategy from non-emptiness witness.

Complexity: non-elementary

Reason: A_φ is of non-elementary size for spec φ in S1S.

Post-1972 Developments

- Pnueli, 1977: Use LTL rather than S1S as spec language.
- Vardi+Wolper, 1983: Elementary (exponential) translation from LTL to automata.
- Safra, 1988: Doubly exponential construction of tree automata for strategy trees wrt LTL spec (using Vardi+Wolper).
- Rosner+Pnueli, 1989: 2EXPTIME realizability algorithm wrt LTL spec (using Safra).
- Rosner, 1990: Realizability is 2EXPTIME-complete.

Standard Critique

Impractical! 2^{EXPTIME} is a **horrible** complexity.

Response:

- 2^{EXPTIME} is just worst-case complexity.
- 2^{EXPTIME} lower bound implies a doubly exponential bound on the size of the smallest strategy; thus, hand design cannot do better in the worst case.

Real Critique

- Algorithmics not ready for practical implementation.
- Complete specification is difficult.

Response: More research needed!

- Better algorithms
- Incremental algorithms – write spec incrementally