A Revisionist History of Algorithmic Game Theory

Moshe Y. Vardi

Rice University

Theoretical Computer Science: Vols. A and B

van Leeuwen, 1990: *Handbook of Theoretical Computer Science*

- Volume A: algorithms and complexity
- Volume B: formal models and semantics (“logic”)

E.W. Dijkstra, EWD Note 611: “On the fact that the Atlantic Ocean has two sides”

- North-American TCS (FOCS&STOC): Volume A.
- European TCS (ICALP): Volumes A&B

A Key Theme in FOCS/STOC: *Algorithmic Game Theory* – algorithm design for strategic environments

Birth of AGT: The ”Official” Version

NEW YORK, May 16, 2012 – ACM’s Special Interest Group on Algorithms and Computation Theory (SIGACT) together with the European Association for Theoretical Computer Science (EATCS) will recognize three groups of researchers for their contributions to understanding how selfish behavior by users and service providers impacts the behavior of the Internet and other complex computational systems. The papers were presented by Elias Koutsoupias and Christos Papadimitriou, Tim Roughgarden and Eva Tardos, and Noam Nisan and Amir Ronen. They will receive the 2012 Gödel Prize, sponsored jointly by SIGACT and EATCS for outstanding papers in theoretical computer science at the International Colloquium on Automata, Languages and Programming (ICALP), July 9–13, in Warwick, UK.

Three seminal papers

- Koutsoupias&Papadimitriou, STACS 1999: *Worst-case Equilibira* – introduced the "price of anarchy" concept, a measure of the extent to which competition approximates cooperation, quantifying how much utility is lost due to selfish behaviors on the Internet, which operates without a system designer or monitor striving to achieve the "social optimum."

- Roughgarden & Tardos, FOCS 2000: *How Bad is Selfish Routing?* – studied the power and depth of the "price of anarchy" concept as it applies to routing traffic in large-scale communications networks to optimize the performance of a congested network.

- Nisan & Ronen, STOC'99: Algorithmic Mechanism Design: studied classical mechanism design from an algorithmic and complexity-theoretic perspectives.

Worst-Case Rquilibria – Price of Anarchy

Question: What is the worst-case ratio between cooperative (centralized) and non-cooperative (distributed) outcomes in games?

Koutsoupias&Papadimitriou – for a simple routing game: n tasks over m links

- *Upper Bound*: $O(\sqrt{m \log m})$
- *Lower Bound*: $\Omega(\log m / \log \log m)$

Roughgarden & Tardos – a more complex routing game:

- *Tight Bound*: $\frac{4}{3}$

Algorithmic Mechanism Design

Background – Game Theory

- *Equilibria Analysis*: analogous to algorithm analysis
- *Mechanism Design*: analogous to algorithm design

– Design games with desirable equilibria – e.g., second-price sealed-bid auction

Nisan & Ronen – n-distributed task-allocation game

- *Upper Bound*: n-approximation mechanism
- *Lower Bound*: No c-approximation mechanism for $c < 2$.

Computing Equilibria

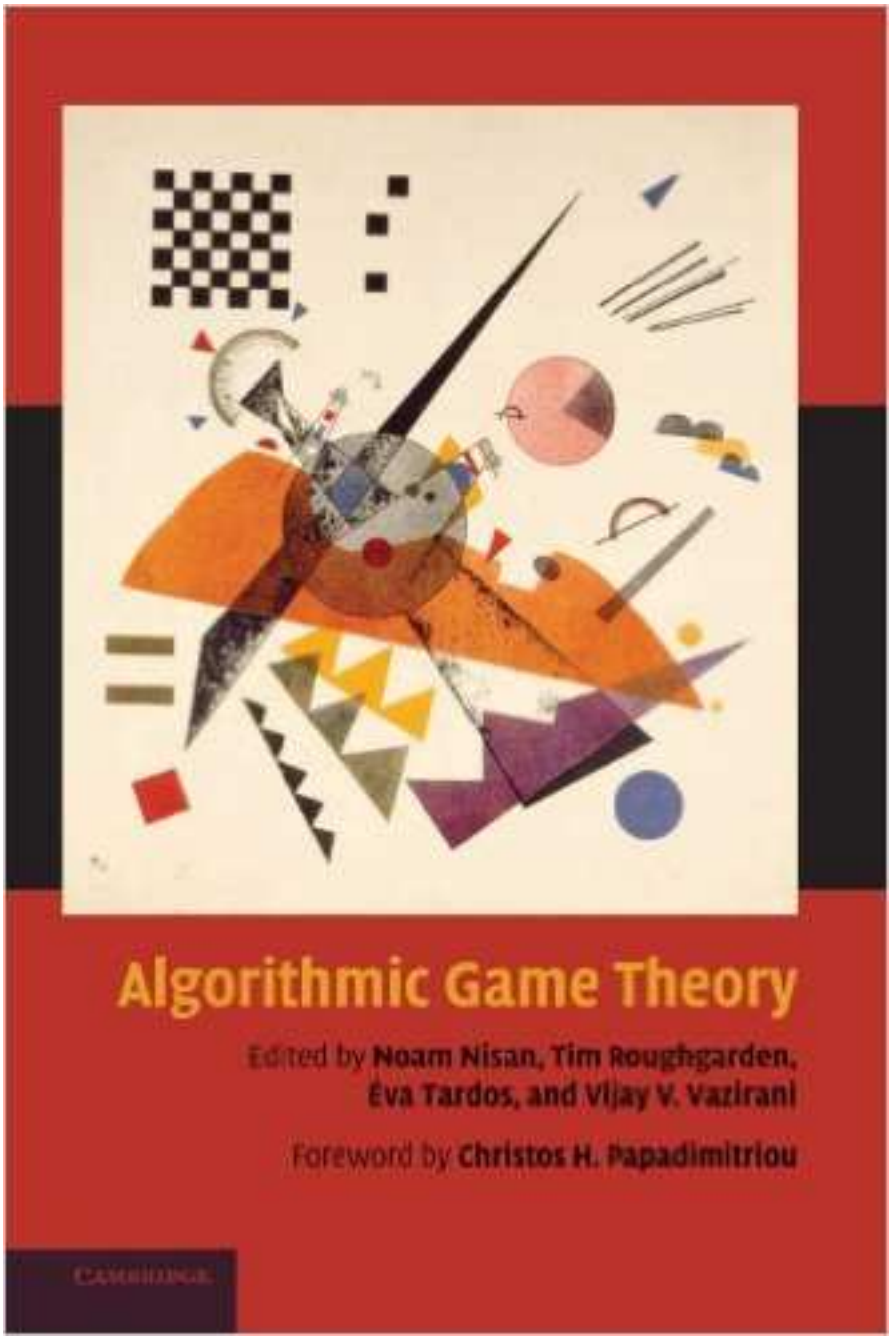
Daskalakis&Goldberg&Papadimitriou, STOC'06: *The complexity of computing a Nash equilibrium* –

- Nash's existence proof is nonconstructive (using Brouwer's Fixed-Point Theorem)
- **Theorem**: Finding a Nash equilibrium in three-players games is PPAD-complete.
- Result extended later to two-players games.

PPAD – polynomial-parity arguments on directed graphs

- Generally believed *not* to be in PTIME for succint graphs.

Algorithmic Game Theory, 2007



A Revisionist Thesis

Thesis:

- AGT was initiated by logicians in 1957!
- AGT was revitalized by Vol. B TCS in 1989!
- AGT has been active in Volume B TCS since then!

Point:

- *First*: Give credit where credit is due!
- *More Significantly*: Tear down the wall between Vol. A and Vol. B!

Gale-Stewart Games

Gale-Stewart Game: an infinite-round, full-information, turn-based, two-player game

- *Finite Action Set* – A, *Winning Set* – $W \subseteq A^\omega$
- Players 0 and 1 alternate choosing actions in A – play is in A^ω
- Player 0 wins if play is in W

Basic Concepts:

- *Strategy*: $f : A^* \to A$
- *Determinacy*: Either Player 0 or Player 1 has a winning strategy.

Question: When is a Gale-Stewart game determined?

- Determinacy holds easily for finite-round games (Zermelo).

Known Results

- Gale&Stewart, 1953: Determinacy for open and closed winning sets
- Martin, 1975: Determinacy for Borel winning sets
- Beyond Borel: depends on set-theoretic axioms

Monadic Second-Order Logic

View an infinite word $w = a_0, a_1, \ldots$ over alphabet A as a mathematical structure:

- *Domain*: $0, 1, 2, \ldots$
- *Dyadic predicate*: $<$
- *Monadic predicates*: $\{P_a : a \in A\}$

Monadic Second-Order Logic (MSO):

- Monadic atomic formulas: $P_a(x)$ $(a \in A)$
- Dyadic atomic formulas: $x < y$
- Set quantifiers: $\exists P, \forall P$

Example: $(\forall y)((\exists x)((y < x)) \wedge P_a(x))$ – infinitely many occurrences of a.

MSO Games

Church's Problem, 1957:

- *Input*: MSO formula φ
- $W = models(\varphi)$ - game is determined!
- *Analysis*: Who wins the game?
- *Synthesis*: Construct a winning strategy for winning player

Solution [Büchi&Landweber, 1969]:

- Analysis problem is decidable – *nonelementary*
- Winning player has a *finite-memory* strategy.
- Finite-memory strategy can be constructed algorithmically.

Rabin, 1972: Simpler solution via *tree automata*.

Claim: This is the birth of AGT!

- An algorithmic approach to a game-theoretical question.
- Algorithm design in a strategic setting.
- *Automated algorithm design*!

But: Nonelementary lower bound [Stockmeyer, 1974].

Finite-Memory Strategies

Transducers [EF Moore, 1956] – $T = (\Sigma, \Delta, S, s_0, \rho, \delta)$

- Σ - finite input alphabet
- Δ - finite output alphabet
- S: finite state set
- $s_0 \in S$: start state
- $\delta : S \times \Sigma \to S$ - transition function
- $\delta : S \to \Delta$ - output function

Extending – $\rho : \Sigma^* \to S$, $\delta : \Sigma^* \to \Delta$.

- $\rho(\epsilon) = s_0$
- $\rho(wa) = \rho(\rho(w), a)$
- $\delta(w) = \delta(\rho(w))$

LTL Games: Rebirth of AGT

Complexity-Theoretic Motivation: Can we get around the nonelementary bound?

Answer: Use *LTL* (Linear Temporal Logic) rather than MSO.

- No explicit quantifiers
- Temporal connectives: *next*, *eventually*, *always*, *until*, *release*
- Expressively equivalent to first-order logic; weaker than MSO.

[Pnueli&Rosner, 1989]: Analysis and synthesis of LTL games is 2EXPTIME-complete.

Linear Temporal Logic

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

Main feature: time is implicit

- *next* φ: φ holds in the next state.
- *eventually* φ: φ holds eventually
- *always* φ: φ holds from now on
- φ *until* ψ: φ holds until ψ holds.

- $\pi, w \models next\ \varphi$ if w 

- $\pi, w \models \varphi\ until\ \psi$ if w 

Examples

Psalm 34:14: “*Depart from evil and do good*”

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

What Good is 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.

Automated Design of Programs

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.

Program Synthesis

The Basic Idea: Mechanical translation of human-understandable task specifications to a program that is known to meet the specifications.

Deductive Approach [Green, 1969, Waldinger and Lee, 1969, Manna and Waldinger, 1980]

- Prove *realizability* of function,
  e.g., $(\forall x)(\exists y)(Pre(x) \rightarrow Post(x, y))$

- Extract *program* from realizability proof.

Classical vs. Temporal Synthesis:

- *Classical*: Synthesize transformational programs

- *Temporal*: Synthesize programs for ongoing computations (protocols, operating systems, controllers, etc.)

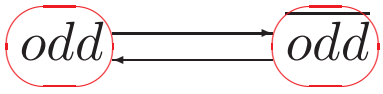
Synthesis of Ongoing Programs

Specs: Temporal logic formulas

Early 1980s: Satisfiability approach
(Wolper, 1981, 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: Ongoing interaction with environment (Harel+Pnueli, 1985), e.g., hardware, operating systems, communication protocols, etc.

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 job is eventually printed
  $always \ \bigwedge_{j=1}^{2}(J_i \rightarrow eventually \ P_i)$

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 is receptive to *all* input sequences.

Conclusion: Satisfiability is inadequate for synthesis of reactive systems.

Realizability

I: input variables, O: output variables

Game:

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

Infinite Play:
$i_0, i_1, i_2, \ldots$
$0_0, 0_1, 0_2, \ldots$

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

Win: behavior $\models$ spec

Specifications: LTL formula on $I \cup O$

Realizability: [Abadi+Lamport+Wolper, 1989 Dill, 1989, Pnueli+Rosner, 1989]
Existence of winning strategy for system.

Synthesis [Pnueli+Rosner, 1989]:
Extraction of winning strategy for system.

Question: LTL is subsumed by MSO, so what did Pnueli and Rosner do?
Answer: better algorithms!

Post-1972 Developments

Pnueli, 1977 : Use LTL rather than MSO as spec language

V.+Wolper, 1983 : Exponential translation from LTL to Büchi automata

Safra, 1989 : Exponential determinization of Büchi automata

Pnueli+Rosner, 1989 : 2EXPTIME realizability algorithm wrt LTL spec

Rosner, 1990 : Realizability is 2EXPTIME-complete.

Post 1990: Emergence of *Parity Games*

Distributed Games

[Pnueli&Rosner, 1990]: Can we synthesize *distributed* protocols?

- Reduce to multi-player games
- Model communications via a directed graph
- *Critical factor*: complete vs. incomplete informaton games

Result: Complexity goes from 2EXPTIME to nonelementary to undecidable.

- Finkbeiner&Schewe, 2005: characterization of decidability

Related Work: multiple-player games with incomplete information [Peterson&Reif, 1979]

Büchi Automata

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

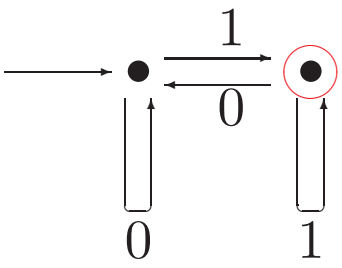
- *Alphabet*: Σ
- *States*: S
- *Initial states*: $S_0 \subseteq S$
- *Transition function*: $\rho : S \times \Sigma \to 2^S$
- *Accepting states*: $F \subseteq S$

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 ω-*Reg* of ω-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 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.

Complexity: nonelementary [Stockmeyer, 1974]

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.

Ongoing Research: practically effective compilation

Determinization

- *Key Insight*: Fundamental mismatch between automaton nondeterminism and strategic behavior – *determinization* required!
- *Obstacle*: Büchi automata are not closed under determinization.

Deterministic Parity Automaton:
$A = (\Sigma, S, S_0, \rho, F)$
- *Alphabet*: Σ
- *States*: S
- *Initial states*: $S_0 \subseteq S$
- *Transition function*: $\rho : S \times \Sigma \to S$
- *Accepting Condition*: $\pi :\to \{0, \ldots, k-1\}$ – priority function

Acceptance: minimal priority visited infinitely often is even.

Determinization Theorem [Safra,1988]: Given an n-state Büchi automaton A, we can construct an equivalent deterministic parity automaton A^d with $n^{O(n)}$ states and n priorities.

Ongoing Research: determinization constructions

Parity Games

Key Ideas in LTL Games: from logic to automata, from automata to graph games

Crux: Many synthesis and model-checking problems are reducible to parity games.

Parity Games $G = (V_0, V_1, E, \pi)$

- $V = V_0 \cup V_1$, $E \subseteq V^2$ – total relation
- Priorities: $\pi : V \to \{0, \ldots, k-1\}$
- Play 0 moves pebble from V_0, Play 1 moves pebble from V_0.
- Pebbles move along edges in E
- *Play*: Sequence in V^ω
- *Winning*: Player i wins if minimum priority visited infinitely often has parity i.

Determinacy: game is determined!

- *Analysis*: Who wins from a given node v of a parity game G?
- *Synthesis*: construct winning strategy for winning player.

Parity Games – Known Result

Emerson&Jutla, 1991: Winner has a *memoryless* (i.e., *history-free*) strategy.

Parameters: n nodes, m edges, k priorities

Complexity:

- Jurdzinski, 1998: in UP∩co-UP
- Schewe, 2007: $O(mn^{\frac{k}{3}})$
- Jurdzinski&Paterson&Zwick, 2008: randomized algorithm in $O(n^{\sqrt{n}})$
- Friedmann&Hansen&Zwick, 2011: Subexponential lower bound for a particular LP algorithm for PGs

Major Open Question: Are parity games solvable in PTIME?

Mean-Payoff Games

[Ehrenfeucht&Mycielski, 1979]: **Mean-Payoff Games** – $G = (V_0, V_1, E, \pi)$

- *Payoff of a play*: *LimAvg* of sum of priorities
- Player 0 maximizes, Player 1 minimizes

Optimization Problem: Compute the maximal value that Player 0 can guarantee at a given node.

Known:

- Jurdzinski, 1998: decision problem in UP∩co-UP
- Björklund&Sandberg&Vorobyov, 2003: randomized subexponential algorithm

A Bridge: Parity games are reducible to mean-payoff games, which are reducible to simple stochastic games [Jurdzinski, 1998, Zwick&Paterson, 1996]

Graph Games

Observation: Analysis of graph games is a vibrant research direction of Volume B TCS!

- *Major Theme*: Probabilistic Behavior

Example – Markov Decision Processes: *Games against Natute*

- Player 1 is stochastic: probabilities on edges

Two Versions:

- *Qualitative Analysis*: analyze for win with probability 1 or > 0
- *Quantitative Analysis*: compute win probability for optimal strategy

Results:

V., 1985 : qualitative analysis in PTIME.

Courcoubetis&Yannakakis, 1989 : quantitative analysis in PTIME.

Incomplete Information

Incomplete Information: One or both players have imperfect (but correct) sensors

Example: Parity Games with incomplete information

- Analysis and synthesis are EXPTIME-complete: follows from [Reif, 1984].

Combining Probability with Incomplete Information: Stochastic Games with Incomplete Information (following [Shapley, 1953])

Results for Stochastic Games:

- *Complete Information*: NP∩co-NP (Chatterjee&Jurdziski&Henzinger, 2004)
- *One Player with Incomplete Information*: undecidable *qualitative* analysis [Baier&Bertrand&Größer, 2008]
- *One Player with Incomplete Information*: undecidable *quantitative* analysis, even for finite-memory strategies (follows from Paz, 1971); even *approximate quantitative* analysis is undecidable (Madani&Hanks&Condon, 2003)
- *One Player with Incomplete Information*: EXPTIME-complete *qualitative* analysis for finite-memory strategies (Chatterjee&Doyen&Nain&V., 2014)

Synthesis from Components

Basic Intuition: [Lustig+V., 2009]

- In practice, systems are typically not built from scratch; rather, they are constructed from existing components.
  - *Hardware*: IP cores, design libraries
  - *Software*: standard libraries, web APIs
  - *Example*: mapping application on smartphone – location services, Google maps API, graphics library
- Can we automate "construction from components"?

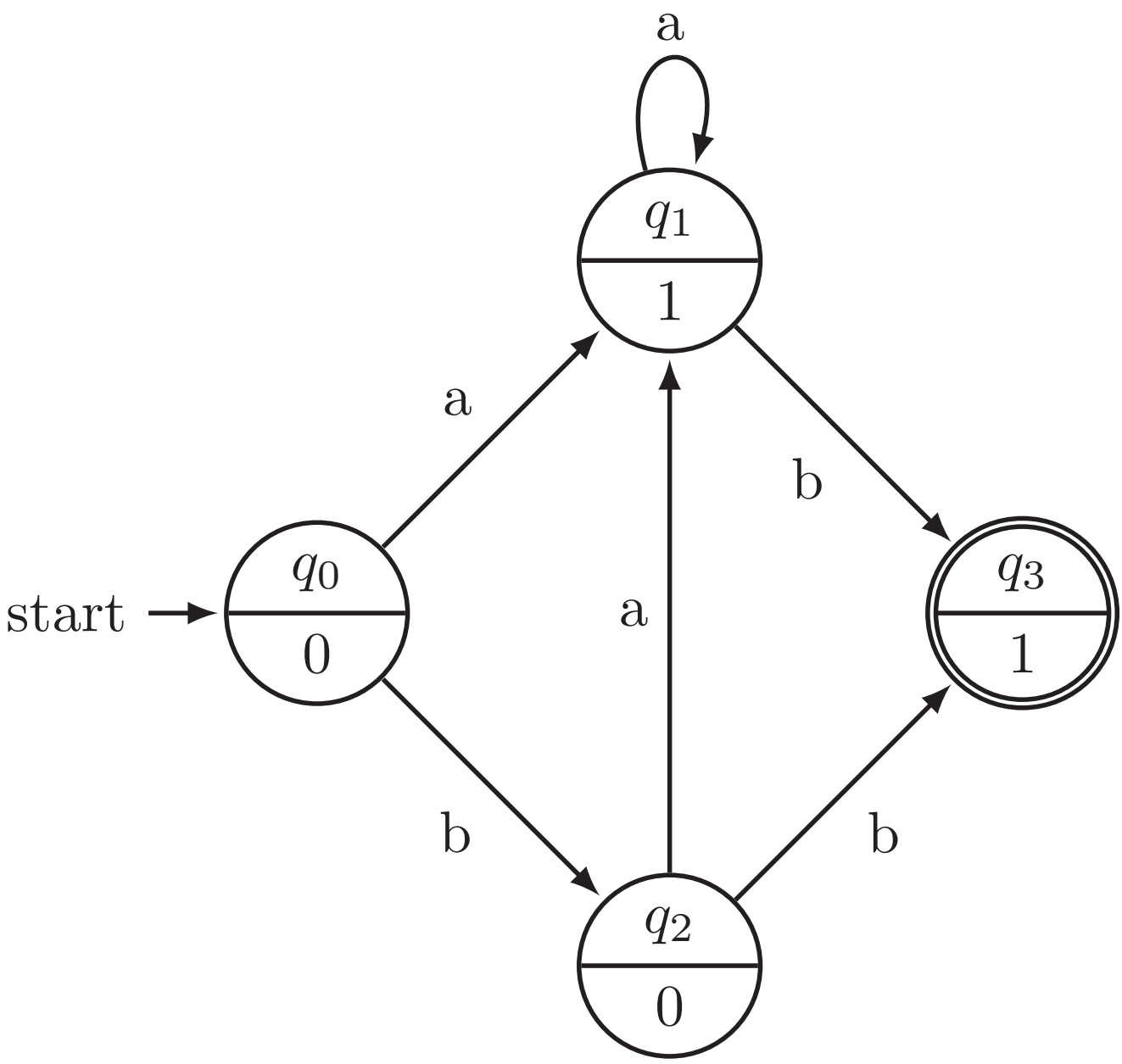
Setup:

- *Library* $L = \{C_1, \ldots, C_k\}$ of *component types*.
- *LTL specification*: φ

Problem: Construct a finite system S that satisfies φ by composing components that are *instances* of the component types in L.

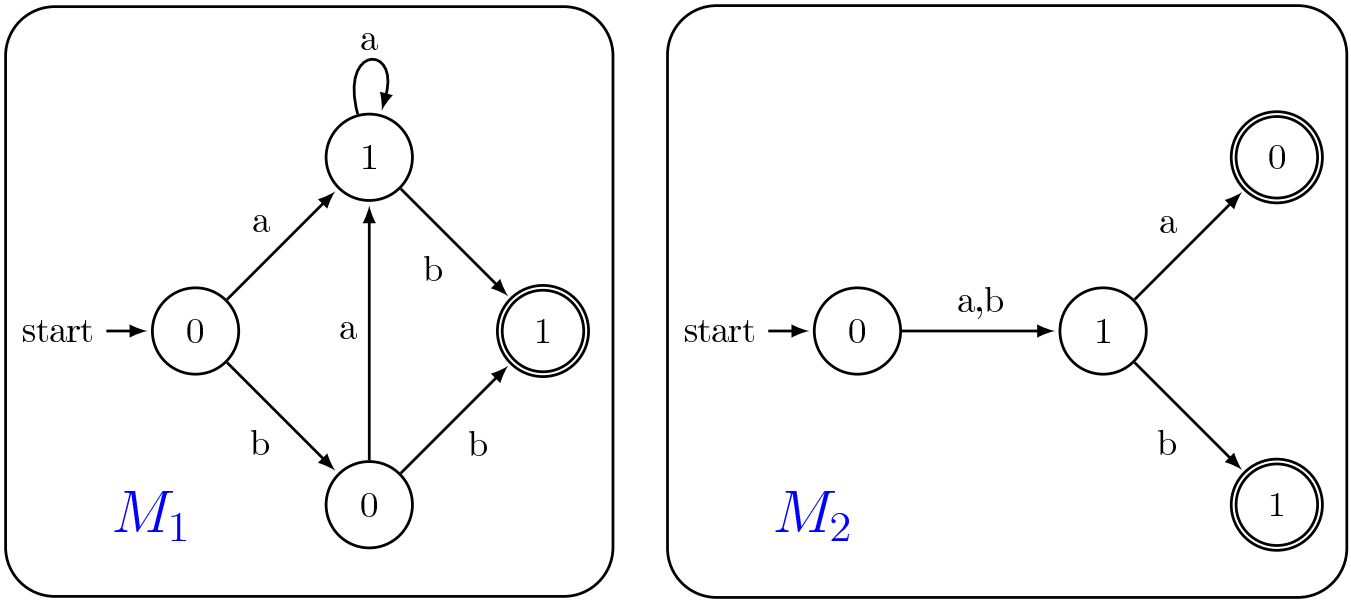
Question: What are components? How do you compose them?

Components: Transducers with Exits



Control-Flow Composition, I

Motivation: Software-module composition – exactly one component interacts with environment at one time; on reaching an exit state, *goto* start state of another component.

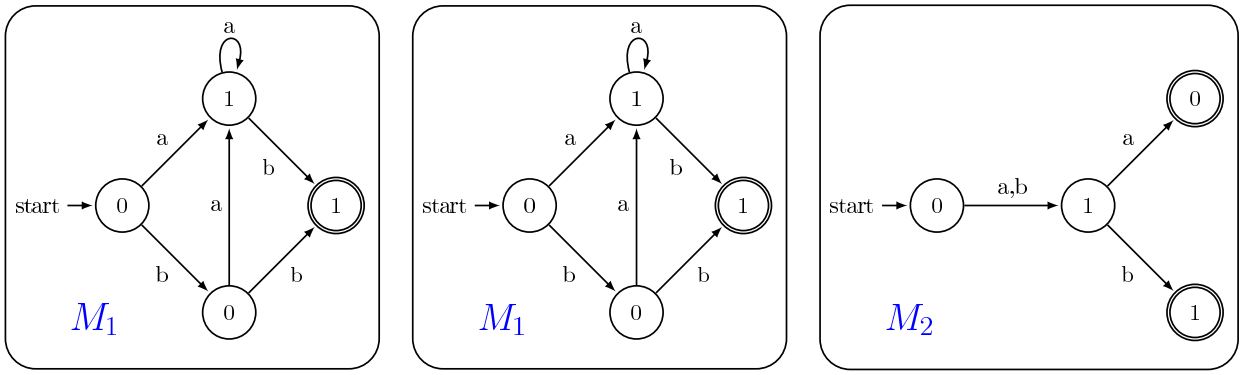


A library of two components types:

$L = \{M_1, M_2\}$

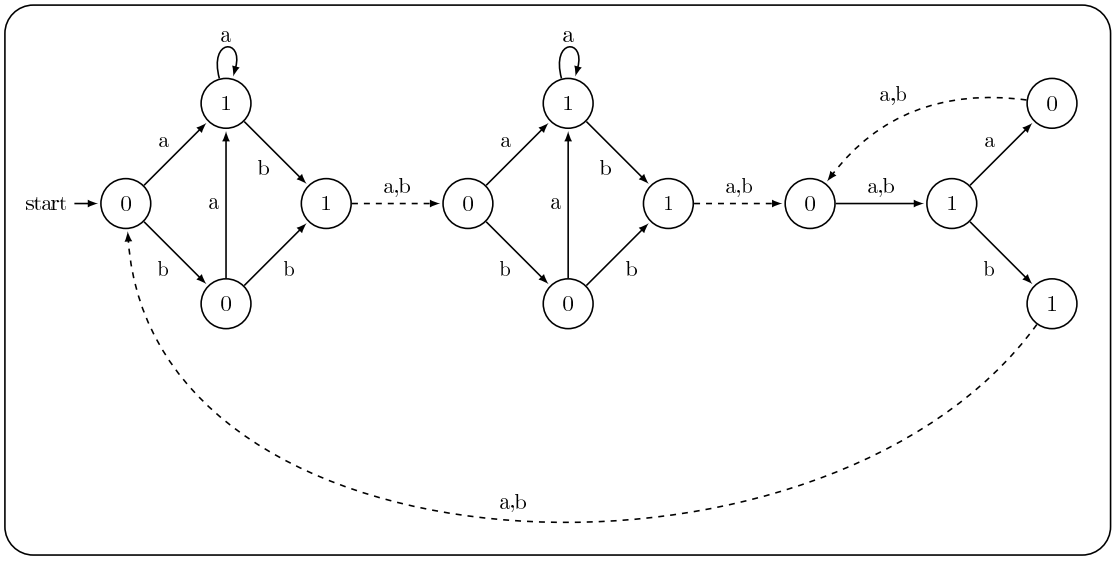
Control-Flow Composition II

Pick three component instances from L:



Control-Flow Composition III

Connect each exit to some start state – resulting composition is a transducer and is *receptive*.



Control-Flow Synthesis

Setup:

- *Components*: transducers with *exit states*, e.g., software module
- *Controlflow composition*: upon arrival at an exit state, *goto* start state of another component – composer chooses target of branch.
- *No* a priori bound on number of component instances!

Theorem: [Lustig+V.,2009]
Controlflow synthesis from components is 2EXPTIME-complete.

Crux:

- Consider general (possible infinite) composition trees, that is, unfoldings of compositions
- Use tree automata to check that all possible computations wrt composition satisfy φ
- Reduce to parity games
- Show that if general composition exists then finite composition exists.

Volume B AGT

Summary:

- Rich nearly-60-years-old theory
- Deep connections between logic, automata, and games
- Fundamental motivation from SW/HW engineering
- *Key*: focus on algorithms and complexity

Mr. Gorbachev, Tear Down That Wall!

Vol. A and Vol. B AGT: Different but equal!

Vol. A → Vol. B:

- Complexity-theoretic approach to parity games
  - *Parity Games Hypothesis (PGH)*: PG$\notin$PTIME‘
  - Consequences of PGH?
- Computing equilibria (e.g., “Rational Synthesis”, by Fisman&Kupferman&Lustig, 2010)
- Price of anarchy in quantitative synthesis, approximations, etc.

Vol. B → Vol. A:

- Extend from equilibria analysis to dynamic analysis (e.g, “Network-Formation Games with Regular Objectives” by Avni&Kupferman&Tamir, 2014).

- Automation of mechanism design (unpublished by Chaudhuri&Fang&V.)