Rational Synthesis

Dana Fisman
Orna Kupferman and Yoad Lustig
Introduction to Formal Verification of Reactive Systems

Rational Synthesis

Some Concepts from Game Theory

Dana Fisman
Orna Kupferman
Yoav Lustig
What are reactive systems?

- Let's start with what they are not...

**Arithmetic problems:**
**Input:** $x, y$.
**Output:** $x+y$.

**Decision problems:**
**Input:** Graph $G$, two vertices $s$ and $t$.
**Output:** Is vertex $t$ reachable from $s$?

**Compilers:**
**Input:** source code.
**Output:** machine code.
What are reactive systems?

Transformational Systems

Input → Processing → Output

Elevator

Reactive Systems

Input → Processing → Output

System → Outer world/Environment
A Simple Reactive System: An Elevator

- The elevator moves among the floors, stops at certain floors and open/closes doors
- in response to the buttons pressed within the elevator and at the different floors
What are reactive systems?

Reactive Systems

Operating system

Web browser

Mail client

Elevator

ATM machine

Cellular phone

System

Outer world/Environment

Input

Processing...

Output
A Simple Reactive System: An Elevator

- The elevator moves among the floors, stops at certain floors and open/closes doors in response to the buttons pressed within the elevator and at the different floors.

How do we reason about such systems?

There is no point in time where we can stop and ask whether the system behaved correctly.
A Simple Reactive System: An Elevator

- Specification:
  - If the button $i$ was pressed then the elevator eventually stops at the $i$th floor.
  - If button $j$ was pressed before button $k$ and the elevator is now at floor $i$ and $i < j < k$ then the elevator will stop at the $j$th floor before it stops at the $k$th floor.

Such specifications are conveniently phrased using Temporal Logic.
Verification

Both are defined over the corresponding inputs and outputs.
A computation of $S$ defines an infinite word $i_0 o_0 i_1 o_1 i_2 o_2 i_3 o_3 \cdots$

A formula $\varphi$ holds/does not hold on such an infinite word.

The system $S$ defines a set of infinite words $L(S)$.

The formula $\varphi$ defines the set of infinite words $L(\varphi)$.

The computations of $S$ are contained in the models of $\varphi$, i.e., $L(S) \subseteq L(\varphi)$.

Model Checking
Model Checking

The automata theoretic approach

- Complexity (LTL): time polynomial in the size of the system, exponential in the size of the formula.

Use automata over **infinite** words

A word (computation) witnessing the non-emptiness provides a counter example

\[ \emptyset \]
Synthesis

- The automatic construction of a system out of its specification.
- The system should produce correct outputs in response to every possible sequence of inputs!

Why bother to build a system and only then verify it?
Can't one automatically built a system that is correct by construction?

- If such a system exists we say that it realizes the specification.
- If no such system exists the specification is unrealizable.
Representing the realizing system

- We can represent a system by full tree whose
  - directions are the inputs
  - nodes are labeled by outputs

- Each path of the tree corresponds to a possible sequence of inputs, and the respective output responses

- If there exists such a tree where the formula holds on all paths then the formula is realizable.
Suppose we have an infinite tree satisfying the specification

Who says there is a finite transducer implementing this tree?
Synthesis can also be seen as a game between the system and the environment

- The system makes a move then the environment and so on
- The system’s objective is to satisfy the specification. The environment’s objective is the opposite
The specification is realizable if there is a winning strategy for the system.

- As it means the system can cope with any input sequence.

Results in the theory of $\omega$-regular games guarantee that if there is a winning strategy – there is one with finite memory.

A strategy with finite memory is a transducer.
The automata theoretic approach:

- Complexity (LTL): doubly-exponential time in the size of the formula.

\( A_{\forall \varphi} \) accepts all trees all of whose (infinite) paths satisfy \( \varphi \).

A tree \( T \) witnessing the non-emptiness of \( A_{\forall \varphi} \) defines a transducer implementing \( \varphi \).
Synthesis Limitations

- What do you do when the specification is unrealizable?
  - Either refine the specification.
  - Or pose limitations/requirements on the environment.
Rational Synthesis

Dana Fisman
Orna Kupferman
Yoad Lustig
Synthesis - weakness of standard approach

- Modern systems often interact with other systems

The standard approach abstracts the way in which the environment is composed of its underlying agents.

The actions of the agents fall into the universally quantified input signals, and there is an implicit assumption that the system should satisfy its specification no matter how the agents behave.

As if the agents conspire to fail the system - (hostile env.)
Our question is: Can system synthesizers capitalize on the rationality and goals of other agents interacting with the system?
Each agent is interested in downloading, but has no incentive to upload.

An agent can download only if the other agent uploads.
Peer-to-peer network (2 agents)

- Formally, for each $i \in \{\text{Alice, Bob}\}$, Agent $i$ controls the bits
  - $u_i$ - Agent $i$ tries to upload
  - $d_i$ - Agent $i$ tries to download.

- The objective of Alice is

  always eventually $(d_{\text{Alice}} \land u_{\text{Bob}})$. 

Property is not realizable since it poses requirements on the inputs.
peer-to-peer example

- Assume Alice declares and follows the following strategy (known as tit-for-tat):
  - I will upload at the first time step: $u_{Alice}(0) := True$
  - And from that point onward I will reciprocate the actions of Bob: $u_{Alice}(k) := u_{Bob}(k - 1)$

- Against this strategy, Bob can only ensure his objective by satisfying Alice’s objective as well.

- Thus, assuming Bob acts rationally, Alice can ensure her objective.

Thus, a synthesizer can capitalize on the rationality of agents involved!
We would like to generate a protocol for the system and each of the agents such that:

- If all follow the protocol, the system’s speciation is met.

and

- The agents have no incentive to deviate from the protocol (assuming they are rational).
How can one formally define rationality?

What does it mean that an agent has no incentive to deviate from the protocol?

Such questions are studied in game theory, and algorithmic mechanism design.
Overview

- Introduction to formal verification
  - Reactive systems
  - Verification
  - Synthesis

- Rational Synthesis
  - What we want (roughly)
  - Some concepts from game theory
  - Rational Synthesis formal definition
  - How to solve the rational synthesis problem
A game arena is a tuple

\[ G = \langle V, v_0, I, (\Sigma_i)_{i \in I}, (\Gamma_i)_{i \in I}, \delta \rangle \]

\( \Gamma_i : V \rightarrow 2^{\Sigma_i} \) specifies the allowed actions for Player \( i \) at each node

\( \Sigma_{\text{White}} \):

- Pawn \( a \) can move
- Pawn \( b \) can move
- Knight \( b \) can move
- Knight \( g \) can move

\( \Sigma_{\text{Black}} \):

- Pawn \( c \) can move
- Pawn \( d \) can move
- Knight \( c \) can move
- Knight \( d \) can move

\( \delta \) is the transition function.
Strategies and related notions

- A strategy of agent $i$: a rule determining which action to take in each possible situation
  
  $$\pi_i: \{\text{situations}\} \rightarrow \{\text{actions}\}.$$  

- A set of strategies, one per each agent is termed a strategy profile $\pi = (\pi_1, \ldots, \pi_n)$

- A strategy profile determines the outcome of the game (when all players follow their assigned strategy)
Strategies and related notions

- With each outcome there is an associated **payoff** for each of the agents. The **payoff** is usually a real number which the player aims to maximize.

- In our setting the **goals** are **temporal specifications**, so the **payoff** is 1 if the specification is met and 0 otherwise.

- A strategy profile \( \pi = (\pi_1, \ldots, \pi_n) \) is termed a **solution concept**, if the players have **no incentive** to deviate from \( \pi \).

- Roughly speaking, they will **not deviate** if a **deviating does not increase** their payoff. Let see some specifics....
Objective of each player is to visit infinitely often a node label by his initial.
A dominant strategy $\pi_i^*$ is a strategy that a player can never lose by adhering to, regardless of the strategies of the other players.

A strategy profile $\pi = (\pi_1, \ldots, \pi_n)$ is in dominant strategies equilibrium if each strategy $\pi_i$ is a dominant strategy.
Dominant strategies - Example

Indeed, in many games not all agents have a dominant strategy, and so a dominant strategy equilibrium may not exists.

<table>
<thead>
<tr>
<th>Player</th>
<th>Dominant strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>$b_2$</td>
</tr>
<tr>
<td>Charlie</td>
<td>both $c_1$ and $c_2$</td>
</tr>
<tr>
<td>Alice</td>
<td>none</td>
</tr>
</tbody>
</table>
Solution Concepts - Nash

- Profile $\pi=(\pi_1,...,\pi_n)$ is a Nash equilibrium if for every player $i$ strategy $\pi_i$ is the best response of player $i$ to the strategies $(\pi_j)_{j\neq i}$ of the other player.

- In other words, $\pi=(\pi_1,...,\pi_n)$ is a Nash equilibrium if a player gains nothings by unilaterally deviating from $\pi$. 
Nash equilibrium exists in almost every game.
Nash equilibrium - Another Example

- It is **irrational** for Bob to stick to his strategy if Alice has deviated from hers!
- Nash **propels nothing** on the case where agents deviate from their strategy!
Solution Concepts - SPE

- $\pi = (\pi_1, \ldots, \pi_n)$ is in **subgame-perfect equilibrium (SPE)** if it forms a **Nash equilibrium from every possible history** (including those non-reachable by $\pi$).
### SPE equilibrium - Example

<table>
<thead>
<tr>
<th>Player</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>$a_2$</td>
</tr>
<tr>
<td>Bob</td>
<td>$b_2$</td>
</tr>
<tr>
<td>Charlie</td>
<td>$c_2$</td>
</tr>
</tbody>
</table>

Diagram:

- **Alice** can choose between $a_1$ and $a_2$.
- **Bob** can choose between $b_1$ and $b_2$.
- **Charlie** can choose between $c_1$ and $c_2$.

The strategy profile is $(a_2, c_2, b_2)$.
Overview

- Introduction to formal verification
  - Reactive systems
  - Verification
  - Synthesis

- Rational Synthesis
  - What we want (roughly)
  - Some concepts from game theory
  - Rational Synthesis formal definition
  - How to solve the rational synthesis problem
Rational Synthesis Problem

- **Given LTL formulas** $\psi, \varphi_1, \varphi_2, \ldots, \varphi_n$ (specifying the objectives of the system and the other agents) and a solution concept $\gamma$ (DS, Nash, SPE, or other)

- **Return a strategy profile** $\pi = (\pi_0, \pi_1, \ldots, \pi_n)$ in the induced game $G$ such that both
  - $\text{outcome}(\pi) \models \psi$
  - the strategy profile $(\pi_1, \ldots, \pi_n)$ is a solution with respect to the solution concept $\gamma$, in the game $G_{\pi_0}$ induced by the system adhering to $\pi_0$.

If such exists...
Rational Synthesis Problem

- If exists such a profile we say that the specification

  \[ \psi, \varphi_1, \varphi_2, \ldots, \varphi_n \]

  is **rationally-realizable** with respect to solution concept \( \gamma \)

- Otherwise we say it is

  **rationally-irrealizable**

  (with respect to solution concept \( \gamma \))
Solution Idea

- We can represent a profile of strategies by strategy-profile trees (next slides)
- We can check that the profile of strategies meets the requirements imposed by rational synthesis using tree automata

\[
\psi, \phi_1, \phi_2, \ldots, \phi_n, \gamma
\]

\[
\varnothing = \varnothing
\]
Strategy Tree - Standard Approach
For Alice and Bob:

A single obedient path

Labeling: Strategies' advice

Branching: Actions taken

<table>
<thead>
<tr>
<th>Player</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>a, A</td>
</tr>
<tr>
<td>Bob</td>
<td>b, B</td>
</tr>
</tbody>
</table>
Starting to Solve

- Using **tree automata** we can check for example that
  - $\varphi$ holds on the **single obedient path**.
  - $\psi_i$ holds on **every path in which only player $i$ deviated from his strategy and all other players adhere to this strategy**.
  - and so on

- **Given a tree automaton (TA) $A_1$ and TA $A_2$** we can build the TA $A_1 \cap A_2$ accepting $L(A_1) \cap L(A_2)$
  - and so on

- **In this way we can obtain a tree automaton accepting only the strategy profiles we seek for**

- **An emptiness check will provide us with a desired strategy profile**
Towards a generic solution

- **Strategy Logic** [CHP07] is a logic that treats strategies in games as explicit first-order objects.

- Can express: DS and Nash
  Cannot express: SPE.

- In order to express SPE we enhance it with first order variables that range over arbitrary histories of the game.
Extended Strategy Logic (ESL)

ESL Syntax:

\[ \Psi ::= \psi(z) \mid \psi(z; h) \mid \Psi \lor \Psi \mid \neg \Psi \mid \exists z_i. \Psi \mid \exists h. \Psi \]

- \( \psi(z) \) – the LTL formula \( \psi \) holds on the single path where all players follow their strategy in \( z=(z_1,\ldots,z_n) \).
- \( \psi(z,h) \) – the LTL formula \( \psi \) holds on the single path where starting when history \( h \) ends all players follow their strategies in \( z=(z_1,\ldots,z_n) \).
- \( \exists z_i. \Psi \) – there exists a strategy \( z_i \) such that \( \Psi(\ldots,z_i,\ldots) \) holds.
- \( \exists h. \Psi \) – there exists a history \( h \) such that \( \Psi(\ldots,h,\ldots) \) holds.

- \( \varphi \) – LTL formula
- \( z = (z_1,\ldots,z_n) \) is a tuple of strategy variables
- \( h \) is a history variable
Expressing Solution Concepts

- Expressing that \( y = (y_1, \ldots, y_n) \) is a DS, Nash or SPE with respect to \( I \) and \( \varphi_1, \varphi_2, \ldots, \varphi_n \)

\[
\psi^{\text{ds}} := \bigwedge_{i \in I} \forall z. (\varphi_i(z) \rightarrow \varphi_i(z_{-i}, y_i))
\]

\[
\psi^{\text{nash}} := \bigwedge_{i \in I} \forall z_i. (\varphi_i(y_{-i}, z_i) \rightarrow \varphi_i(y))
\]

\[
\psi^{\text{spe}} := \bigwedge_{i \in I} \forall z_i. (\varphi_i(y_{-i}, z_i) \rightarrow \varphi_i(y))
\]

All players adhere to \( y_i \) except for player \( i \) which adheres to \( z_i \)
Expressing the solution to the rational synthesis problem given $\gamma \in \{\text{DS, Nash, SPE}\}$

$$\Phi^\gamma := \exists(y_i)_{i \in I}. (\varphi_0((y_i)_{i \in I}) \land \psi^\gamma((y_i)_{i \in I}))$$
We have phrased the problem of rational synthesis in ESL.

If we can determine ESL - that is, given a formula in ESL,
- answer whether it is satisfiable,
- and if it is, find a satisfying assignment

Then, given a rational synthesis problem \( \psi \varphi_1 \varphi_2 \cdots \varphi_n \gamma \)
we can in this manner answer
- whether it is rationally-realizable,
- and if it is, provide a desired strategy profile \( \pi_0 \pi_1 \pi_2 \cdots \pi_n \) as an answer.
Dealing with Arbitrary Histories

- How do we extend Strategy Logic to deal with arbitrary histories?

- A formula $\psi(z,h)$ stipulates that $\psi$ should hold along the path that starts at the root of the tree, goes through $h$ and then follows the profile $z$.

- Thus, adding history variables to strategy logic results in a memoryful logic [KV06]
  - The construction there involves a satellite implementing the subset construction of this automaton.

- Here we use instead strategy-history trees, defined next.
A single path whose **prefix** follows the **history** and whose **suffix** follows the **strategy**
ESL Decidability

- The tree automaton $A_\Psi$ for $[\Psi]$ is defined by induction on the structure of $\Psi$.

$$\Psi ::= \psi(z) \mid \psi(z; h) \mid \Psi \lor \Psi \mid \neg \Psi \mid \exists z_i. \Psi \mid \exists h. \Psi$$

We can build a tree automaton that checks $\phi$ holds on the respective path $A$ over an alphabet $\Gamma \times \Gamma'$.

$A'$ over alphabet $\Gamma$ such that $L(A') = L(A)|_{\Gamma'}$.

$|A'| = 2^{|A| \cdot k}$
**Theorem**

Let $\Psi$ be an ESL formula over a game $G$. Let $d$ be the alternation depth of $\Psi$.

We can construct an APT $A_\Psi$ accepting $[\Psi]_G$ whose emptiness can be checked in time $(d+1)$-EXPTIME in $|\Psi|$.
Rational Synthesis

Theorem

The **LTL rational-synthesis problem** is **2EXPTIME-complete** for the solution concepts of

- dominant,
- Nash equilibrium, and
- subgame-perfect equilibrium.

Same complexity as traditional LTL synthesis.
Conclusion

- Defined & solved synthesis that considers an environment composed of rational agents.

- Can salvage non realizable specifications.  
  We do NOT limit the environment.  
  Instead, we capitalize on the environment goals and rationality!

- Complexity: $2\text{EXPTIME}$-complete (as standard synthesis).

- We didn’t see: an extension for multi-valued setting.
Discussion

- A limitation of our work is that it assumes a **fixed** number of players, the specification for each of them is know.

- It is desirable to extend it to a **parameterized setting**, where the number of players can vary and the specification is parameterized.
The End

- Questions?

- Thank you!
Thank you!

The End