From Formulas to Systems

this is a formula

this is a black box

Orna Kupferman
Hebrew University

this is a system
Is the system correct?
Is the system correct?

1960+

Simulation-based Verification
Simulation-based Verification

Execute the system in parallel with a reference model...

...with respect to some input sequences.

exhaustive?
Is the system correct?

1960+
Simulation-based Verification

1980+
Formal Verification
Formal Verification:

System $\rightarrow$ A mathematical model $M$

Desired behavior $\rightarrow$ A formal specification $\psi$

The system has the desired behavior

$M$ satisfies $\psi$

Model checking
Model checking:

A mathematical model of the system:

A formal specification of the desired behavior:

“every request is followed by a grant”

“only finitely many grants”

...
Temporal logic

• Atomic propositions: \( AP = \{ p, q, \ldots \} \)
• Boolean operators: \( \neg, \land, \lor, \ldots \)
• Temporal operators:
  - \( G \) (always)
  - \( F \) (eventually)
  - \( X \) (next)
  - \( U \) (until)

\[
\begin{align*}
Gp & \quad \begin{array}{l}
\end{array} \\
Fp & \quad \begin{array}{l}
\end{array} \\
Xp & \quad \begin{array}{l}
\end{array} \\
pUq & \quad \begin{array}{l}
\text{p p p p p p p p p p p p p p q} \\
\text{p p p p p p p p p p p p p p q}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\psi_1 &= G \ (req \rightarrow F \ grant) \\
\psi_2 &= GF \ grant \\
\psi_3 &= req \ U (\neg req \lor grant)
\end{align*}
\]
Model checking:

Input: a system $M$ and a specification $\psi$.

Output: does $M$ satisfy $\psi$?

Fully automatic.

State-explosion problem

Counter example

It Works!

symbolic methods, compositionality, abstraction
It’s hard to design systems:
Synthesis:

**Input:** a specification $\psi$.

**Output:** a system satisfying $\psi$.

WOW!!!
Synthesis:

**Input:** a specification $\psi$.

**Output:** a system satisfying $\psi$.

**Input:** $p \land q$.

**Output:** $p,q$ truth assignment for $p \land q$.
Satisfiability of temporal logic specifications:

Is \( Gp \land F \neg p \) satisfiable?

A state of the system: \( \sigma \in 2^{AP} \)

A computation of the system: \( \pi \in (2^{AP})^\omega \)

A specification: \( L \subseteq (2^{AP})^\omega \)

specifications \( \rightarrow \) languages
The automata-theoretic approach:

An LTL specification $\psi$.

\[ L(A_\psi) = \{ \pi : \pi \text{ satisfies } \psi \} \]

Specifications describe infinite computations $\Rightarrow$ we need automata on infinite words.

Büchi 1962: reduce decidability of monadic second order logic to the nonemptiness problem of automata on infinite words.
finite words: the run ends in an accepting state
$L(A) = (a+b)*b$

infinite words: the run visits an accepting state infinitely often
$L(A) = (a*b)^\omega$
Büchi automata

dualization:

finite words: the run ends in an accepting state
$L(A) = (a+b)^*b$  \(L(\tilde{A}) = \varepsilon + (a+b)^*a = (a+b)^* \setminus L(A)\)

infinite words: the run visits an accepting state infinitely often
$L(A) = (a^*b)^\omega$  \(L(\tilde{A}) = (b*a)^\omega \neq (a+b)^\omega \setminus L(A)\)

Büchi dualization: co-Büchi (visit \(\alpha\) only finitely often)
The automata-theoretic approach:

An LTL specification \( \psi \).

LTL \( \rightarrow \) nondeterministic Büchi word automata

\[ [VW86] \]

An automaton \( A \psi \).

\[ L(A \psi) = \{ \pi : \pi \text{ satisfies } \psi \} \]

\( \psi \) is satisfiable \iff \( A \psi \) is nonempty

LTL \( \rightarrow \) nondeterministic Büchi word automata

\[ \psi = G (\text{req } \rightarrow F \text{ grant}) \]

\( A \psi \):

\{\}, \{\text{grant}\}, \{\text{req, grant}\}

\{\text{req}\}

\{\text{grant}\}, \{\text{req, grant}\}

\{\text{req}, \}\]
An example:

1. Whenever user i sends a job, the job is eventually printed.

2. The printer does not serve the two users simultaneously.

1. $G(j_1 \rightarrow F\ p_1) \land G(j_2 \rightarrow F\ p_2)$

2. $G((\neg\ p_1) \lor (\neg\ p_2))$

Let’s synthesize a scheduler that satisfies the specification $\psi$...
Satisfiability of $\psi$: such a scheduler exists?

NO!

A model for $\psi$: help in constructing a scheduler?

NO!

A model for $\psi$: a scheduler that is guaranteed to satisfy $\psi$ for some input sequence.

Wanted: a scheduler that is guaranteed to satisfy $\psi$ for all input sequences.
Closed vs. open systems

Closed system: no input!

all input sequences = some input sequence

\[ 0_0, 0_1, 0_2, \ldots, 0_i \]
Closed vs. open systems

Open system: interacts with an environment!

\[
\begin{align*}
o_0 & \quad \text{AP} = \mathbb{I} \cup \mathbb{O} \\
o_1 &= f(i_0) \\
o_2 &= f(i_0, i_1) \\
o_3 &= f(i_0, i_1, i_2) \\
\end{align*}
\]

An open system: \( f \cdot (2^\mathbb{I})^* \Rightarrow 2^\mathbb{O} \)
\[ f: (2^I)^* \rightarrow 2^O \] is a **regular strategy** if for all \( \sigma \in 2^O \), the set of words \( w \in (2^I)^* \) for which \( f(w) = \sigma \) is regular.

**Regular strategies \rightarrow Finite-state transducers**
Closed vs. open systems

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

In the printer example: $I=\{j_1,j_2\}, \ O=\{p_1,p_2\}$

$f:(\{\},\{j_1\},\{j_2\},\{j_1,j_2\})^* \rightarrow (\{\},\{p_1\},\{p_2\},\{p_1,p_2\})$
A computation of $f$:

\( f(\epsilon) \rightarrow (i_0, f(i_0)) \rightarrow (i_1, f(i_0, i_1)) \rightarrow (i_2, f(i_0, i_1, i_2)) \rightarrow \ldots \)

The specification $\psi$ is realizable if there is $f: (2^I)^* \rightarrow 2^O$ such that all the computations of $f$ satisfy $\psi$. 
ψ is satisfiable $\iff$ ψ is realizable?

Yes! (for all $\rightarrow$ exists)

ψ is satisfiable $\rightarrow$ ψ is realizable?

NO!
Solving the synthesis problem: [Rabin 70, Pnueli Rozner 88]

Key idea: use automata on infinite trees
Word automata: $M(q_0,a) = \{q_1, q_2\}$

Tree automata: $M(q_0,a) = \{(q_1,q_3), (q_2,q_1)\}$
Trees:

Two parameters:
D: a set of directions (binary trees: D={l,r}).
Σ: a set of labels (Σ = {a,c}).

Σ-labeled D-trees

In the realizability story:
D = 2^I (all possible input sequences)
Σ = 2^{I∪O} (label by both input and output).
Solving the synthesis problem: [Rabin 70, Pnueli Rozner 88]

Given an LTL specification $\psi$ over $I \cup O$:

1. Construct a tree automaton $A_\psi$ on $2^{I \cup O}$-labeled $2^I$-trees such that $A_\psi$ accepts exactly all the trees all of whose paths satisfy $\psi$.

2. Obtain from $A_\psi$ a tree automaton $A'_\psi$ on $2^O$-labeled $2^I$-trees that reads the $I$-component of the alphabet form the direction of the nodes.

A tree accepted by $A'_\psi$:

$$f: (2^I)^* \rightarrow 2^O$$ whose computation tree satisfies $\psi$!

3. Check $A'_\psi$ for emptiness.

(With respect to regular trees)
Solving the synthesis problem: [Rabin 70, Pnueli Rozner 88]

1. Construct a tree automaton $A_\psi$ on $2^{I\cup O}$-labeled $2^I$-trees such that $A_\psi$ accepts exactly all the trees all of whose paths satisfy $\psi$.

- **Determinize** the nondeterministic word automaton for $\psi$ and **expand** it to a tree automaton.

  \[ M_+(q,a) = \langle M(q,a), M(q,a) \rangle \]
  
  $M(q_0,a) = \{q_1\}$
  
  $M_+(q_0,a) = \{\langle q_1, q_1 \rangle\}$

*How to construct $A_\psi$??*
Solving the synthesis problem: [Rabin 70, Pnueli Rozner 88]

Do we really have to determinize the word automaton for \( \psi \)?

\[
\text{expand: } M_\triangle(q,a) = M(q,a) \times M(q,a)
\]

\[
M(q_0,a)=\{q_1,q_2\}
\]

\[
M_\triangle(q_0,a)=\{\langle q_1,q_1 \rangle, \langle q_1,q_2 \rangle, \langle q_2,q_1 \rangle, \langle q_2,q_2 \rangle\}
\]

Does not work! We have to determinize!

The same guess should work for all paths in the same subtree.
Solving the synthesis problem: [Rabin 70, Pnueli Rozner 88]

3. Check \( A'\psi \) for emptiness.

Solving nonemptiness of parity tree automata...

Do we really have to use a richer acceptance condition??

Yes, deterministic Büchi is too weak.

Büchi acceptance: visit \( \alpha \) infinitely often

\[
\begin{array}{c}
0,1 \\
\downarrow \\
s \\
1 \\
\downarrow \\
q
\end{array}
\]

\[ L(A) = (0+1)^* . 1^\omega \]

No deterministic Büchi automaton for \( L(A) \) [Landweber 76]
Solving the synthesis problem: [Rabin 70, Pnueli Rozner 88]

3. Check $A'\psi$ for emptiness.

Solving nonemptiness of parity tree automata...

Do we really have to use richer acceptance condition??

Yes, deterministic Büchi is too weak.

parity acceptance: much more complicated...

$\alpha: Q \to \{1,...k\}$

the minimal color that is visited infinitely often is even

...and complex.
That is too bad!!!

- The determinization construction is very complicated.
  - hard to understand
  - hard to implement
  - complicated data structure (no symbolic implementation)

Model checking: tools! A success story!!

Synthesis: no tools, no story.
Kupferman Vardi 2005: avoid determinization

Given an LTL formula $\psi$:

1. Construct a nondeterministic Büchi word automaton $A_{\neg \psi}$ that accepts all computations satisfying $\neg \psi$. Easy [VW86]

2. Run the dual universal co-Büchi word automaton on the $(2^I)$-tree. Easy, running a universal automaton on a tree is sound and complete.

3. Check emptiness of the universal co-Büchi tree automaton. Easy, translate it to a nondeterministic Büchi tree automaton
The magic:

universal co-Büchi tree automata $\rightarrow$ nonterministic Büchi tree automata

Based on an analysis of accepting runs of co-Büchi automata

A run is accepting iff the vertices of its run DAG can get ranks in \{0,\ldots,k\} so that ranks along paths decrease and odd ranks appear only finitely often.

The nondeterministic automaton: guesses a ranking, checks decrease, checks infinitely many visits to even ranks.
Richer Settings:

1. Synthesis with incomplete information

2. Synthesis of a distributed system

3. Specifications in branching temporal logic
The synthesis challenge:

1. **Complexity**  
   (doubly-exponential in the specification)

2. **Compositional and incremental synthesis**

3. **Richer specification formalisms**

4. **Measuring the quality of a specification**