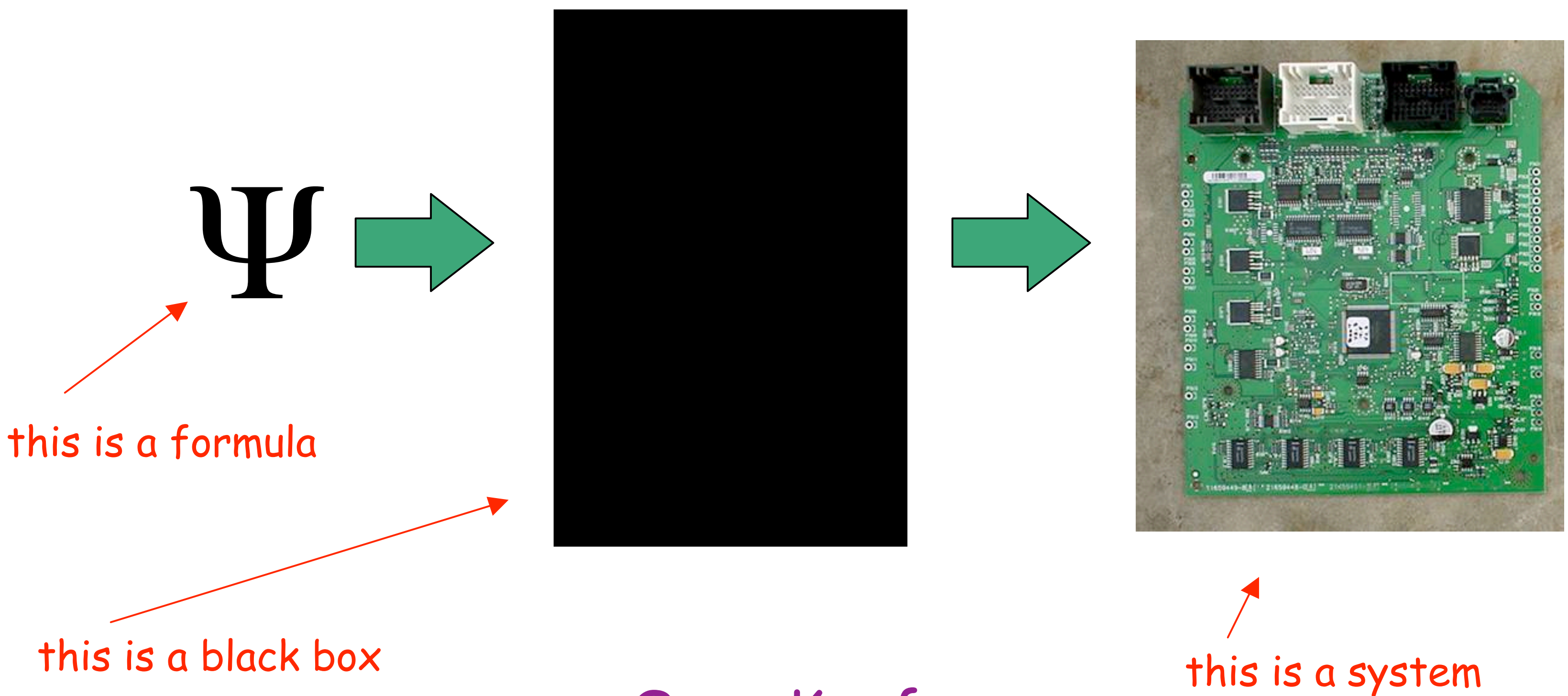
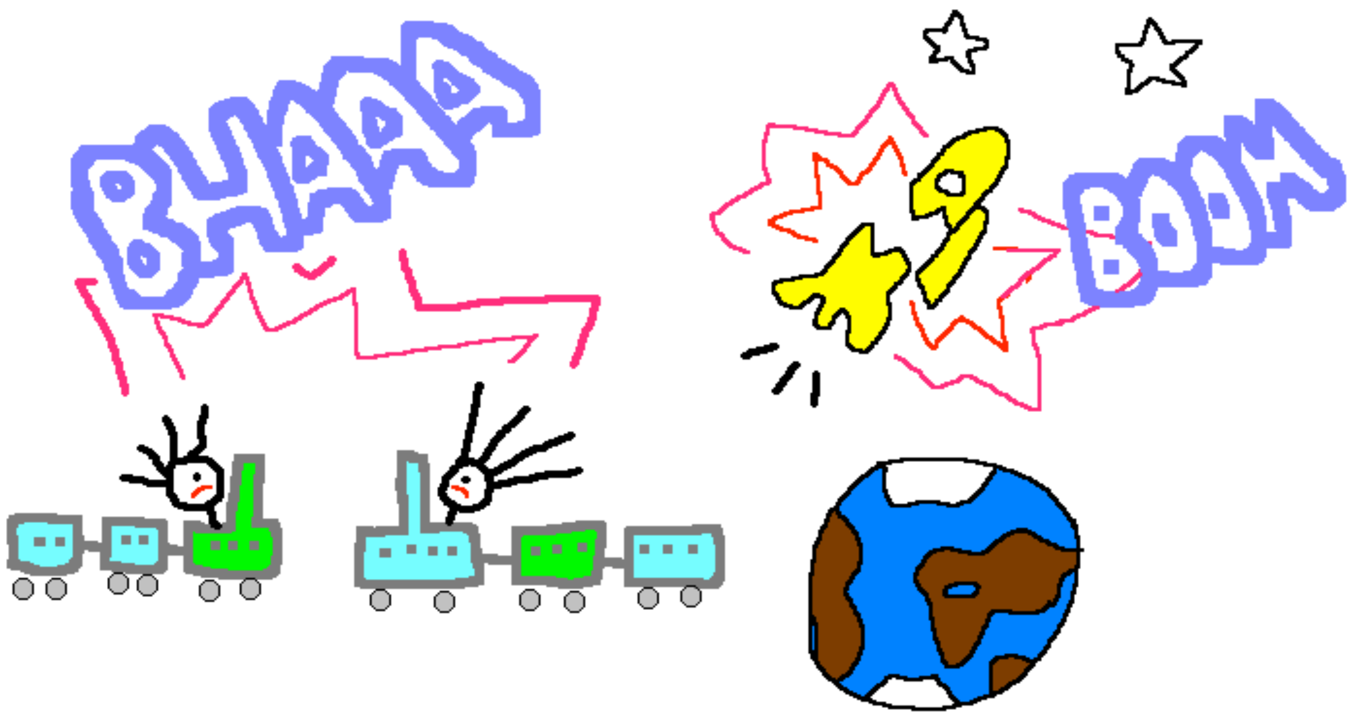
From Formulas to Systems



Orna Kupferman

Hebrew University

Is the system correct?



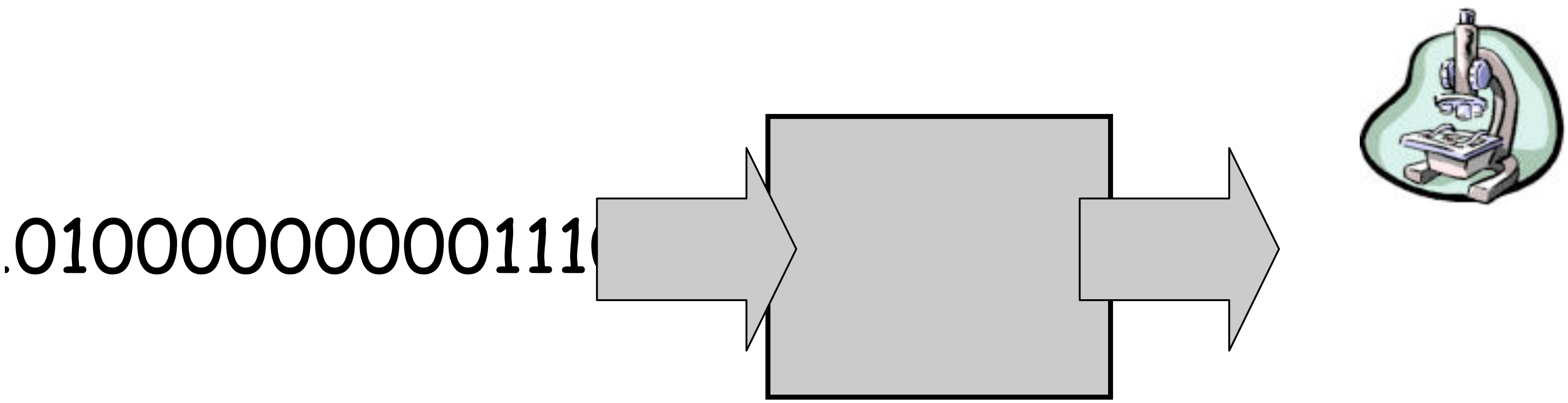
Is the system correct?

1960+

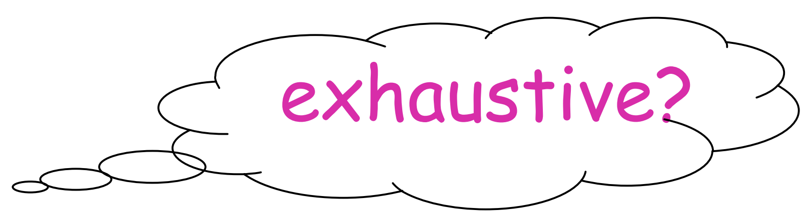
Simulation-based Verification

Simulation-based Verification

Execute the system in parallel
with a reference model…



…with respect to some input sequences.



Is the system correct?

1960+

1980+

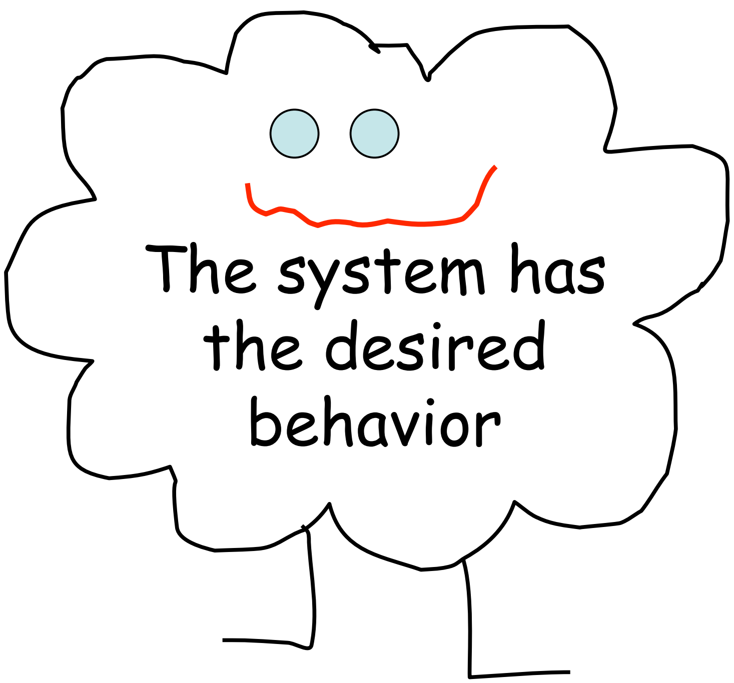
Simulation-based Verification

Formal Verification

Formal Verification:

System → A mathematical model M

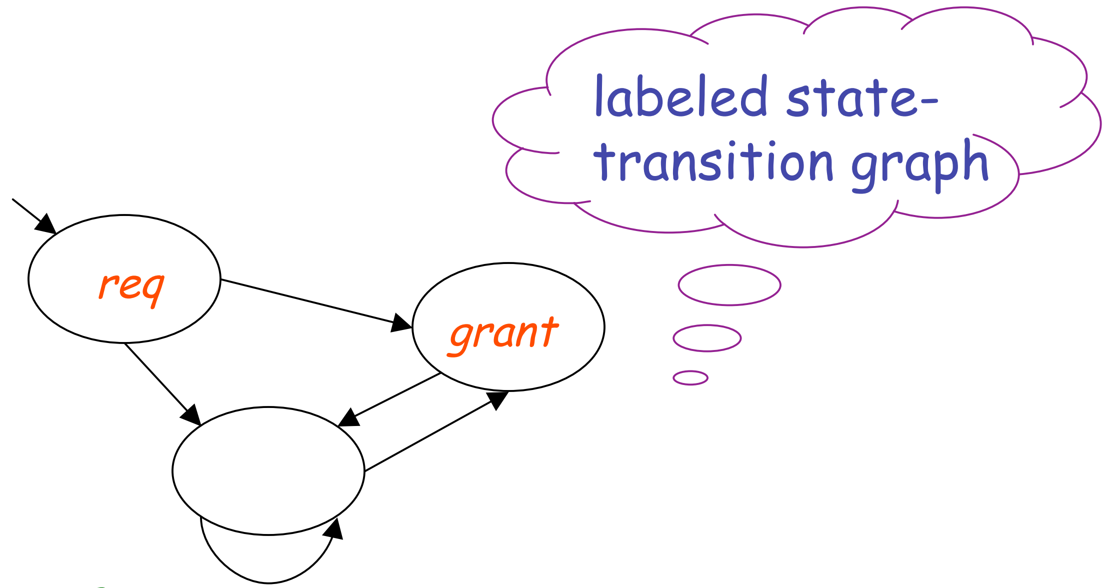
Desired behavior → A formal specification ψ



Model checking

Model checking:

A mathematical model of the system:



A formal specification of the desired behavior:

"every request is followed by a grant"

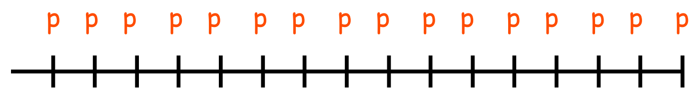
"only finitely many grants"

...

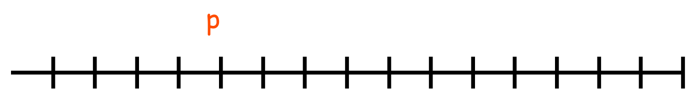
Temporal logic

Church 1957, Prior 1957, Pnueli 1977

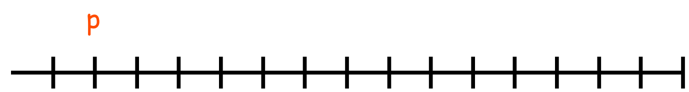
- Atomic propositions: $AP=\{p,q,\ldots\}$
- Boolean operators: $\neg$, $\wedge$, $\vee$, ...
- Temporal operators:
  - G (always) — $\mathrm{G}p$



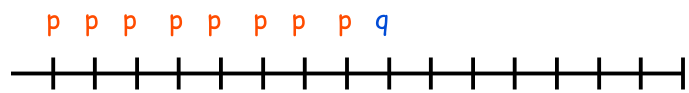
  - F (eventually) — $\mathrm{F}p$



  - X (next) — $\mathrm{X}p$



  - U (until) — $p\,\mathrm{U}\,q$



$\psi_1 = \mathrm{G}\,(req \rightarrow \mathrm{F}\ grant)$

$\psi_2 = \mathrm{GF}\ grant$

$\psi_3 = req\ \mathrm{U}\ (\neg req \vee grant)$

It Works!

symbolic methods, compositionality, abstraction

But...

It's hard to design systems:

Synthesis:

Input: a specification ψ.

Output: a system satisfying ψ.

WOW!!!

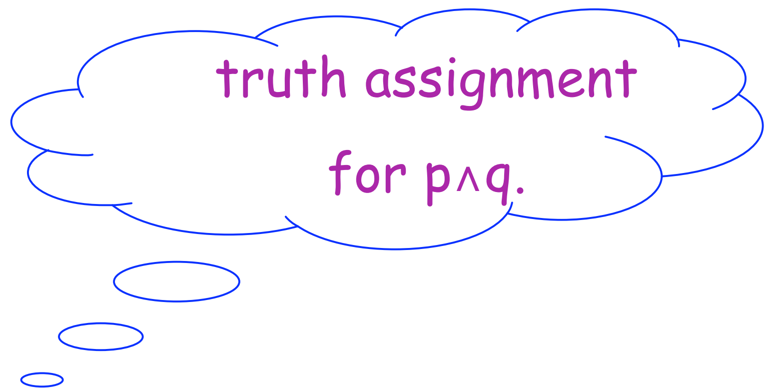
Synthesis:

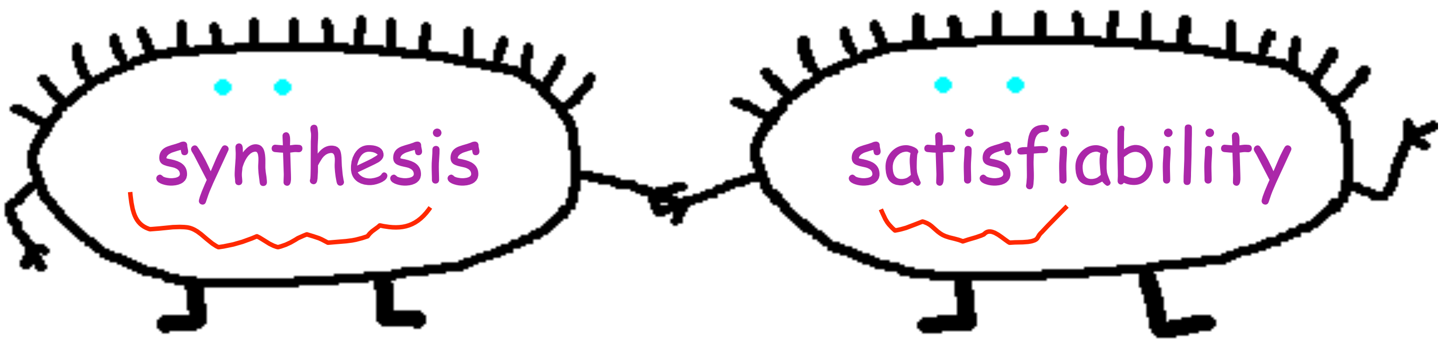
Input: a specification ψ.

Output: a system satisfying ψ.

Input: $p \wedge q$.

Output: p,q

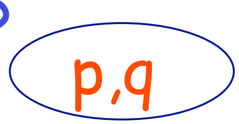




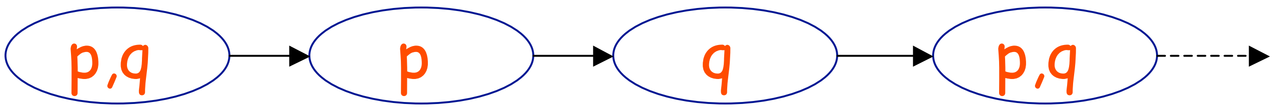
Satisfiability of temporal logic specifications:

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

Is $Gp \wedge F\neg p$ satisfiable?



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



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

specifications → languages

The automata-theoretic approach:

An LTL specification ψ.

⇓

[VW86]

⇓

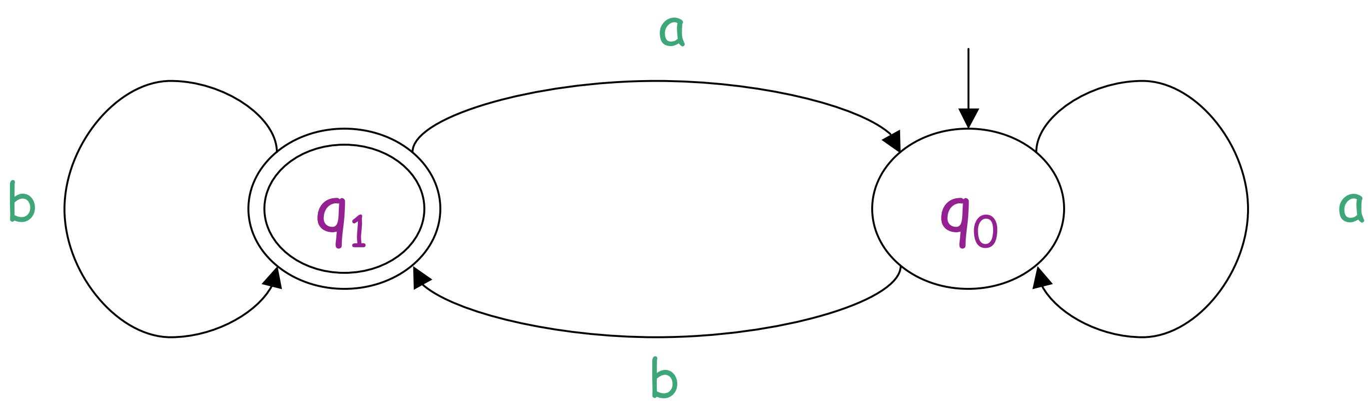
An automaton $A\psi$.

$L(A\psi)=\{ \pi : \pi$ 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.

Büchi automata

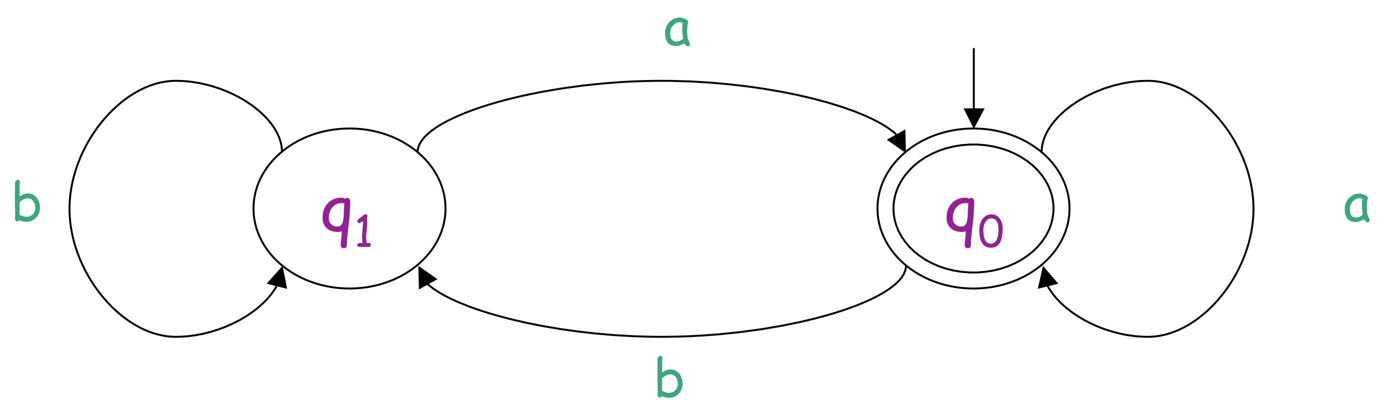


finite words: the run ends in an accepting state
$L(A)=(a+b)^*b$

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 α only finitely often)

The automata-theoretic approach:

An LTL specification ψ.

LTL → nondeterministic Büchi word automata

[VW86]

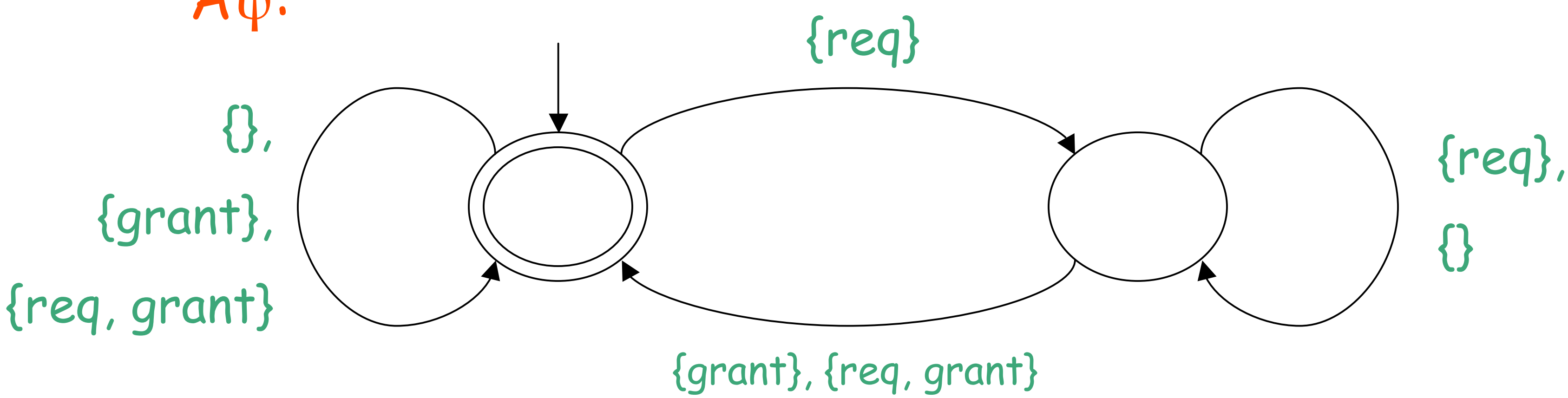
An automaton $A\psi$.

ψ is satisfiable ⇔ $A\psi$ is nonempty

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

$\psi = G\ (req \rightarrow F\ grant)$

$A\psi$:



An example:

user 1

user 2

1. Whenever user i sends a job, the job is eventually printed.
2. The printer does not serve the two users simultaneously.

1. $G(j1 \rightarrow F\ p1) \wedge G(j2 \rightarrow F\ p2)$
2. $G((\neg p1) \vee (\neg p2))$

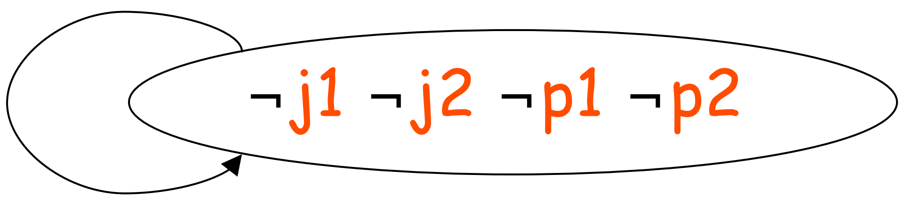
Let's synthesize a scheduler that satisfies the specification ψ…

Satisfiability of ψ ⇒ such a scheduler exists?

NO!

A model for ψ ⇒ help in constructing a scheduler?

NO!

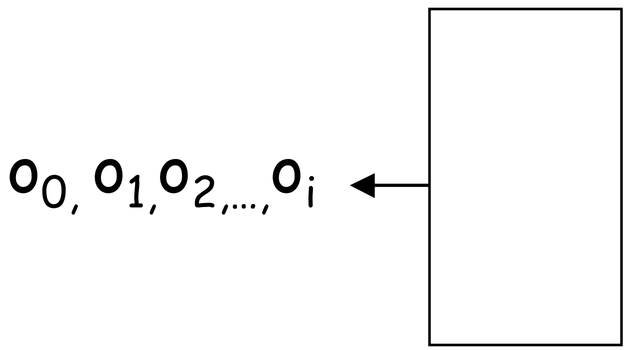


A model for ψ: a scheduler that is guaranteed to satisfy ψ for some input sequence.

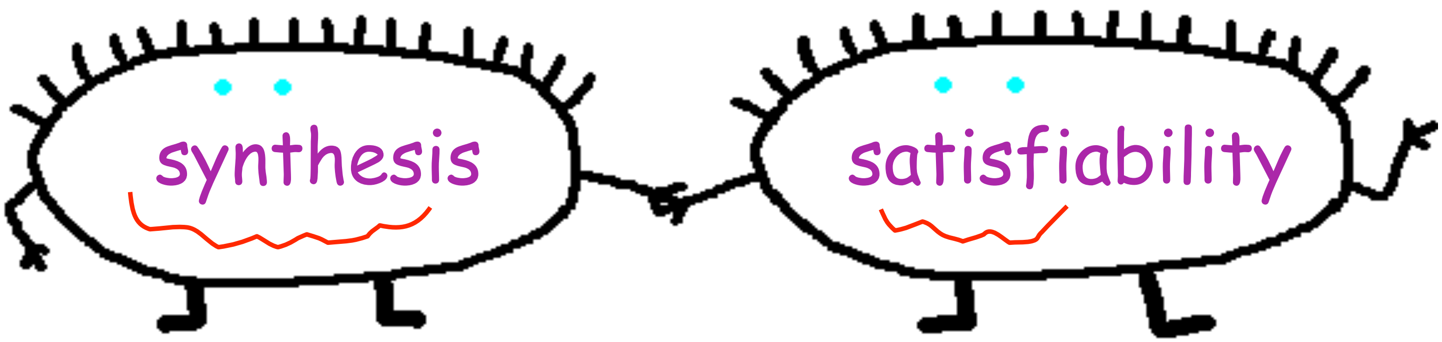
Wanted: a scheduler that is guaranteed to satisfy ψ for all input sequences.

Closed vs. open systems

Closed system: no input!



all input sequences=some input sequence



Closed vs. open systems

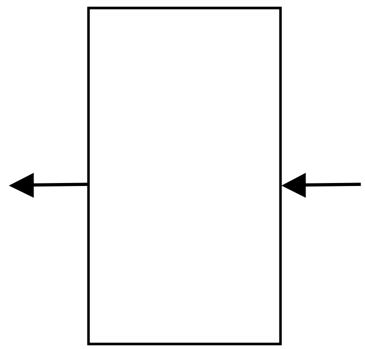
Open system: interacts with an environment!

o_0

$o_1 = f(i_0)$

$o_2 = f(i_0, i_1)$

$o_3 = f(i_0, i_1, i_2)$



i_0

i_1

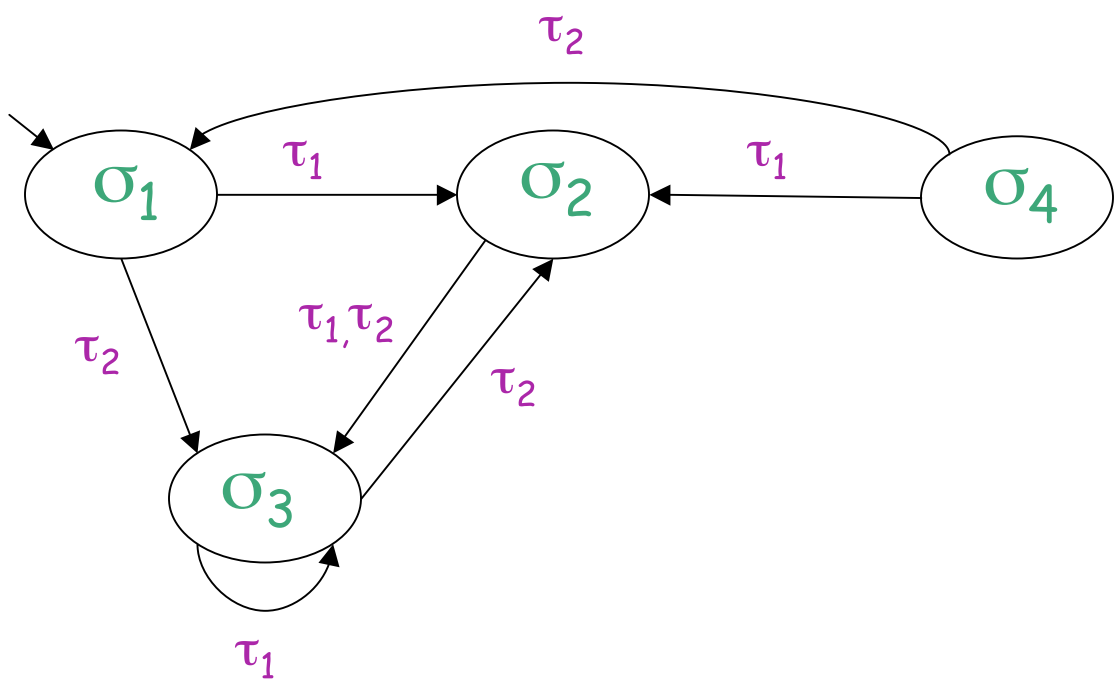
i_2

$AP = I \cup O$

An open system: ~~$f: (2^I)^* \rightarrow 2^O$~~

$f:(2^I)^* \rightarrow 2^O$ 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 → Finite-state transducers

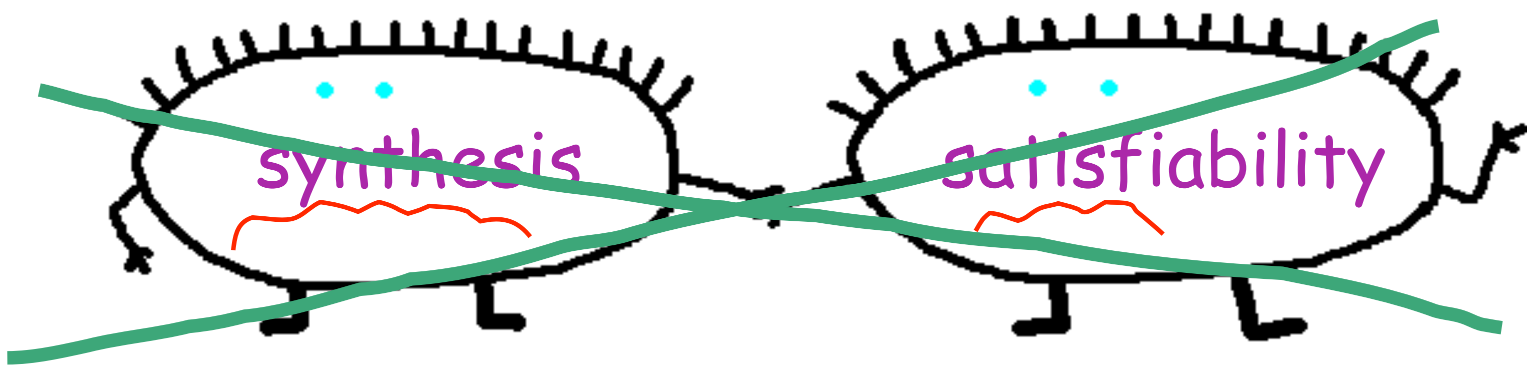


Closed vs. open systems

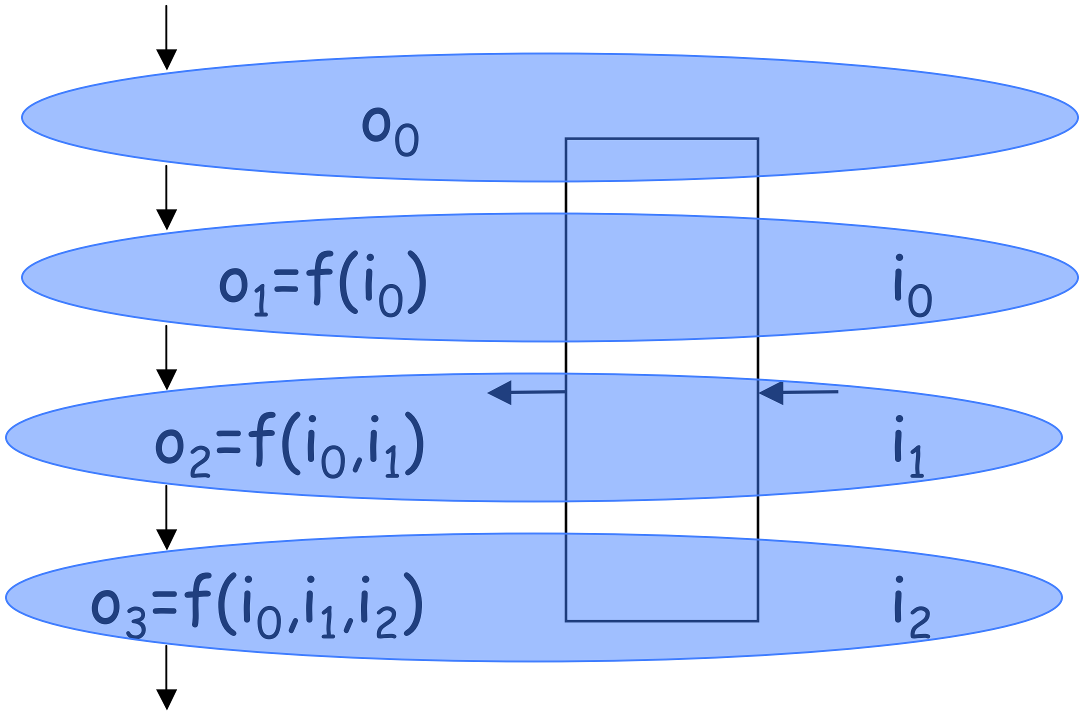
Open system: $f:(2^I)^* \rightarrow 2^O$

In the printer example: I={j1,j2}, O={p1,p2}

$f:(\{\{\},\{j1\},\{j2\},\{j1,j2\}\})^* \rightarrow \{\{\},\{p1\},\{p2\},\{p1,p2\}\}$

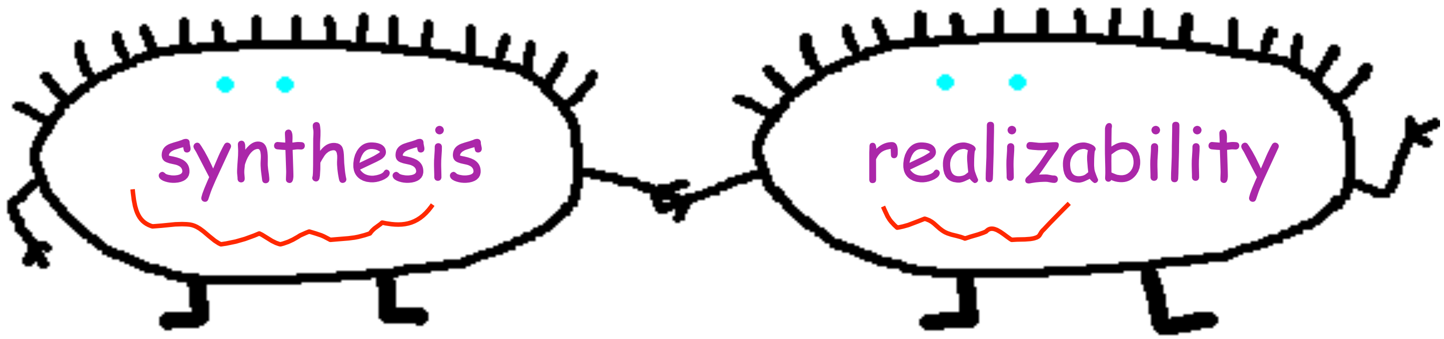


A computation of f:



$(f(\varepsilon)) \rightarrow (i_0,f(i_0)) \rightarrow (i_1,f(i_0,i_1)) \rightarrow (i_2,f(i_0,i_1,i_2)) \rightarrow \ldots$

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



ψ is satisfiable ← ψ is realizable ?

Yes! (for all → exists)

ψ is satisfiable → ψ is realizable ?

NO!

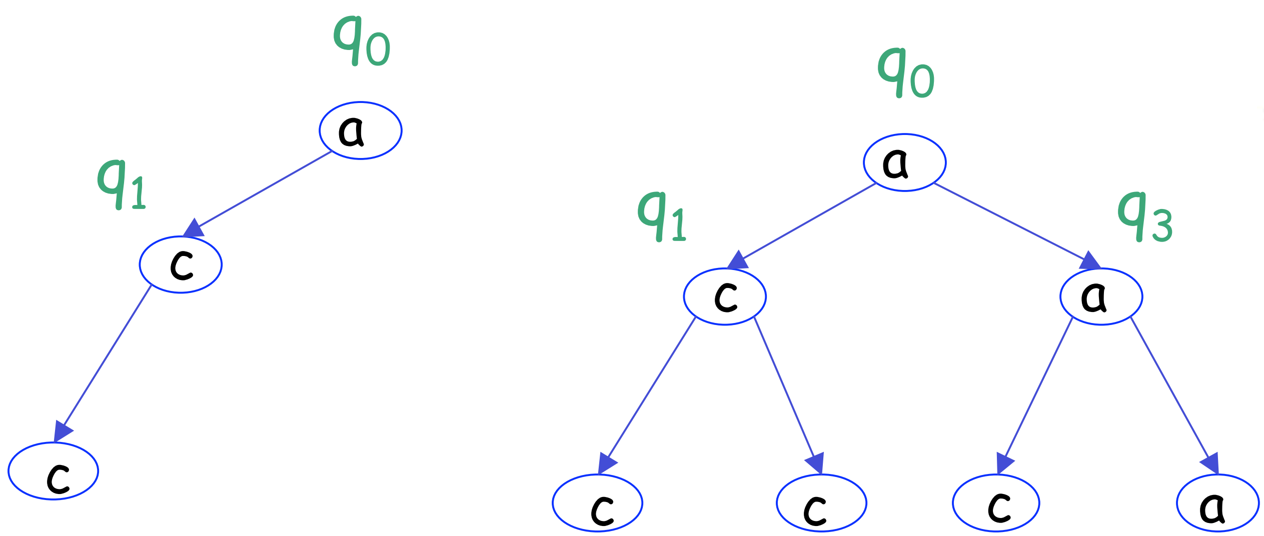
Solving the synthesis problem: [Rabin 70, Pnueli Rozner 88]

Church's problem

Specifications in LTL, improved complexity

Key idea: use automata on infinite trees

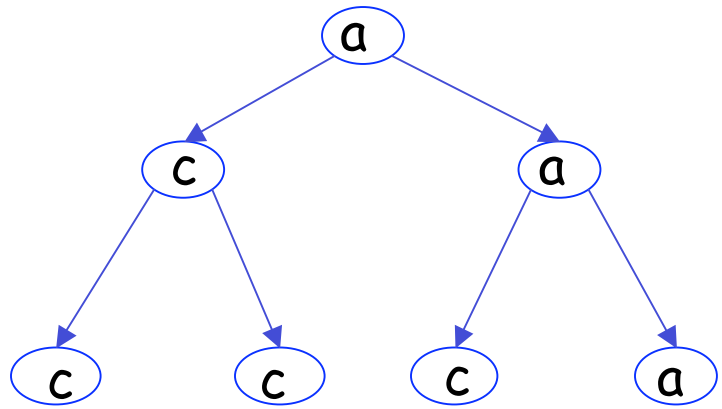
Tree Automata



Word automata: $M(q_0,a)=\{q_1,q_2\}$

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

Trees:



Two parameters:

D: a set of directions (binary trees: D={l,r}).

Σ: a set of labels (Σ = {a,c}).

$f: D^* \rightarrow \Sigma$

Σ-labeled D-trees

In the realizability story:

$D = 2^I$ (all possible input sequences)

$f: (2^I)^* \rightarrow 2^{I \cup O}$

$\Sigma = 2^{I \cup O}$ (label by both input and output).

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

Solving the synthesis problem: [Rabin 70, Pnueli Rozner 88]

Given an LTL specification ψ 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 ψ.
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 ψ!

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 ψ.

How to construct $A\psi$??

- Determinize the nondeterministic word automaton for ψ and expand it to a tree automaton.

expand: $M_t(q,a) = \langle M(q,a), M(q,a) \rangle$

$M(q_0,a)=\{q_1\}$

$M_t(q_0,a)=\{\langle q_1,q_1 \rangle\}$

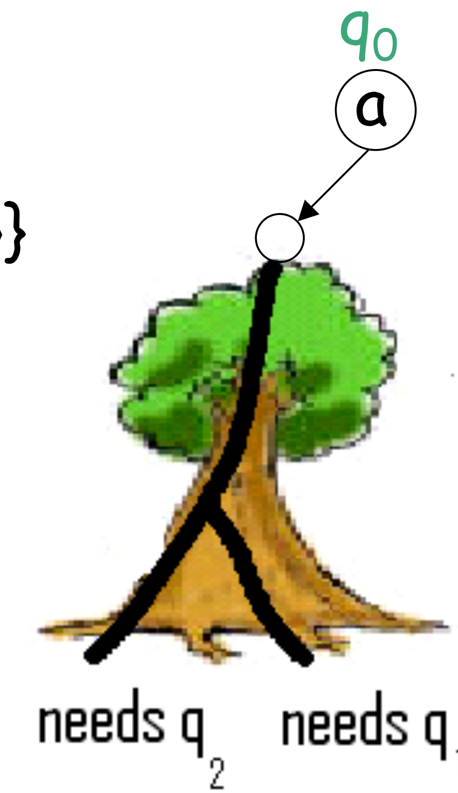
Solving the synthesis problem: [Rabin 70, Pnueli Rozner 88]

Do we really have to determinize the word automaton for ψ?

expand: $M_t(q,a) = M(q,a) \times M(q,a)$

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

$M_t(q_0,a)=\{\langle q_1,q_1\rangle, \langle q_1,q_2\rangle, \langle q_2,q_1\rangle, \langle q_2,q_2\rangle\}$



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]

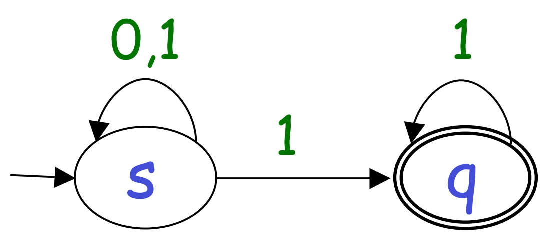
3. Check $A'\psi$ for emptiness.

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 α infinitely often



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

No deterministic Büchi automaton for L(A) [Landweber 76]

Solving the synthesis problem: [Rabin 70, Pnueli Rozner 88]

3. Check A'_ψ 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…

$\alpha: Q \rightarrow \{1,\ldots,k\}$

the minimal color that is visited infinitely often is even

…and complex.

That is too bad!!!

-The determinization construction is very complicated.

[Safra 1988]

- hard to understand
- hard to implement
- complicated data structure (no symbolic implementation)

-Solving parity emptiness is not a big pleasure either.

- deeply nested fixed points
- complicated symbolic implementation

Model checking: tools! A success story!!

Synthesis: no tools, no story.

Kupferman Vardi 2005: avoid determinization

Given an LTL fromula ψ:

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 → nonterministic Büchi tree automata

k depends on the size of the automaton.

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

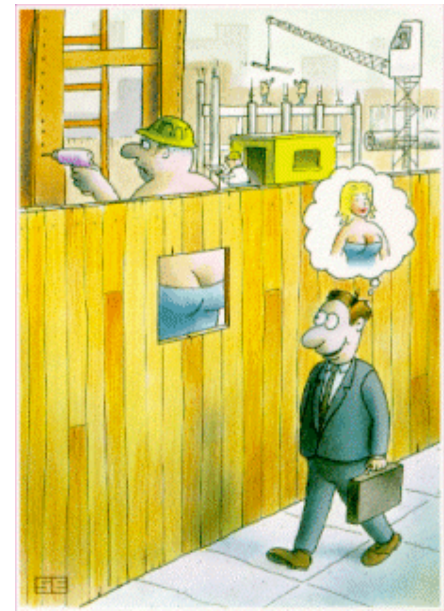
A run is accepting iff the vertices of its run DAG can get ranks in {0,…,k} so that ranks along paths decrease and odd ranks appear only finitely often.

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