BACK TO THE FUTURE

A Fresh Look at Linear Temporal Logic

Javier Esparza
Technical University of Munich

Joint work with Jan Křetínský and Salomon Sickert

Logic

Logic

If all bandersnatches are borogoves
and all borogoves are slithy,
then all bandersnatches are slithy.

Logic

If all bandersnatches are borogoves
and all borogoves are slithy,
then all bandersnatches are slithy.

„True“

Logic

If all X are Y and
all Y are Z,
then all X are Z.

Logic

Logic is the subject of identifying true statements in a language of which you only know a few words.

Temporal Logic

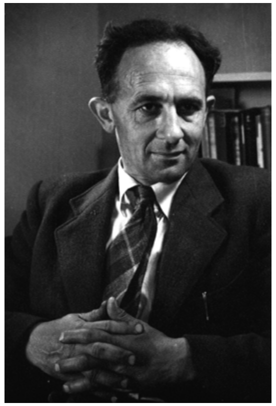
Different logics study different language fragments:

– Propositional logic: and, or, not, if ... then

– Temporal logic: propositional logic + today, tomorrow, eventually, never, ...

Studied within mathematical logic since the end of the XIX century.

Clarence Lewis (1883-1964)



Arthur Prior (1914-1969)

Temporal logic in computer science

- Amir Pnueli proposes in 1977 to use temporal logic to reason about computer programs

אמיר פנואלי

Amir Pnueli

(1941-2009)

Turing Award 1996

THE TEMPORAL LOGIC OF PROGRAMS*

Amir Pnueli
University of Pennsylvania, Pa. 19104
and
Tel-Aviv University, Tel Aviv, Israel

Summary:

A unified approach to program verification is suggested, which applies to both sequential and parallel programs. The main proof method suggested is that of temporal reasoning in which the time dependence of events is the basic concept. Two formal systems are presented for providing a basis for temporal reasoning. One forms a formalization of the method of intermittent assertions, while the other is an adaptation of the tense logic system K_b, and is particularly suitable for reasoning about concurrent programs.

Temporal logic in computer science

A worker that succeeds in acquiring a lock will **eventually** release it, assuming its "doResult" call returns.

The req_close_state is **always** in close_enabled state.

If artist1 registers for event2 **before** artist2 does, then **once** dispatcher receives event2 from the ADT, it will **first** send it to artist1 **and then** to artist2.

The OK button on the login window is enabled **as soon as** the application is started and the login window is **first** displayed to the user.

None of the available methods can be called **until** connect is called.

Mathew Dwyer, Temporal Specification Patterns, https://matthewbdwyer.github.io/psp/

Linear Temporal Logic (LTL)

- LTL extends propositional logic with temporal operators.
- Syntax:

$$\varphi := \mathbf{true} \mid \mathbf{false} \mid p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$$

$$\mathbf{X}\varphi_1 \mid \varphi_1 \mathbf{U} \varphi_2 \mid \varphi_1 \mathbf{W} \varphi_2$$

$\varphi_1 \mathbf{R} \varphi_2 \mid \varphi_1 \mathbf{M} \varphi_2 \mid$ past operators (Past LTL)

$\mathbf{F}\varphi \coloneqq \mathbf{true}\ \mathbf{U}\ \varphi$ (eventually φ or finally φ).

$\mathbf{G}\varphi \coloneqq \varphi\ \mathbf{W}\ \mathbf{false}$ (always φ or globally φ).

Temporal logic in computer science

A worker that succeeds in acquiring a lock will **eventually** release it, assuming its "doResult" call returns.

$$\mathbf{G}(\mathrm{call}_{\mathrm{doResult}} \rightarrow \mathbf{F}\ \mathrm{return}_{\mathrm{doResult}}) \rightarrow \mathbf{G}(\mathrm{return}_{\mathrm{lockacq}} \rightarrow \mathbf{F}\ \mathrm{call}_{\mathrm{lockrel}})$$

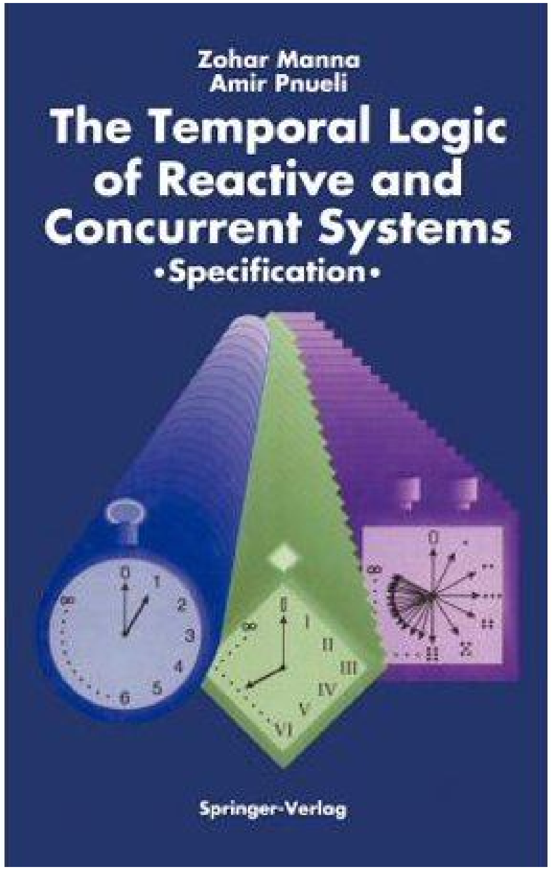
If artist1 registers for event **before** artist2 does, then **once** dispatcher receives event from the ADT, it will **first** notify artist1 **and then** artist2.

$$\mathbf{G}((\mathrm{reg.}a_1 \wedge (\neg \mathrm{unreg.}a_1\ \mathbf{U}\ (\mathrm{reg.}a_2 \wedge (\neg \mathrm{unreg.}a_1 \wedge \neg \mathrm{unreg.}a_2)\ \mathbf{U}\ \mathrm{notify})))$$
$$\rightarrow$$
$$\mathbf{F}\ (\mathrm{notify} \wedge (\neg \mathrm{notify.}a_2\ \mathbf{U}\ \mathrm{notify.}a_1)))$$

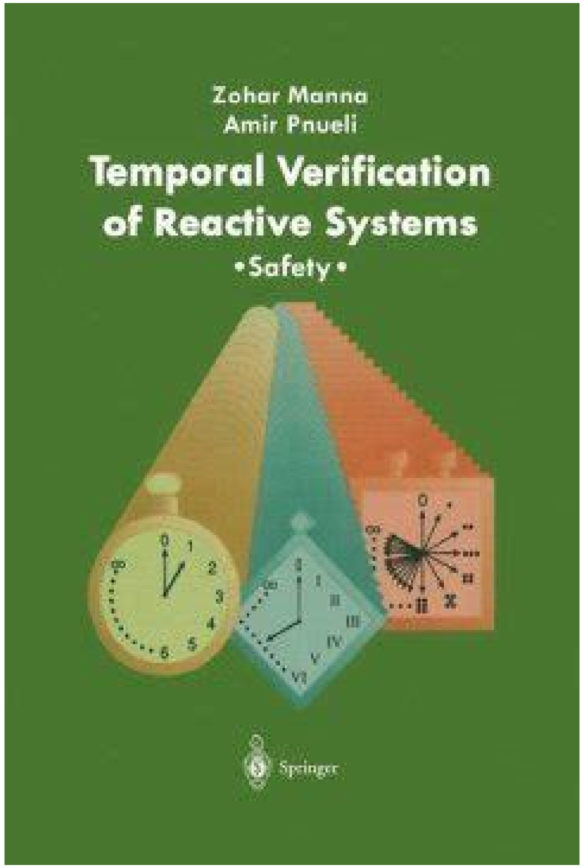
Specifying and verifying reactive systems

זוהר מנה
Zohar Manna
(1939-2018)

אמיר פנואלי
Amir Pnueli
(1941-2009)

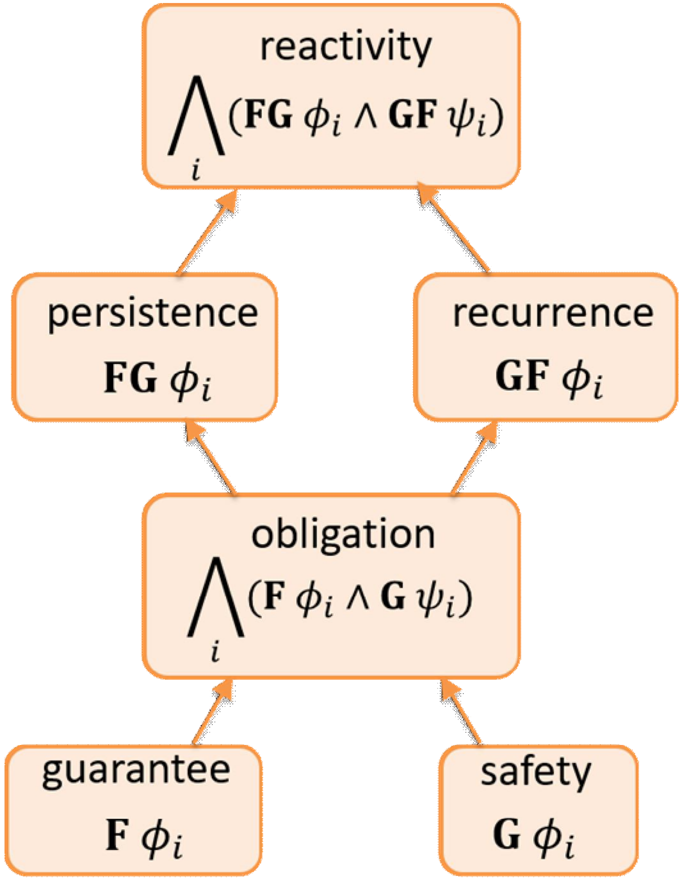


1992



1995

The Safety-Progress Hierarchy



ϕ_i and ψ_i
are past formulas

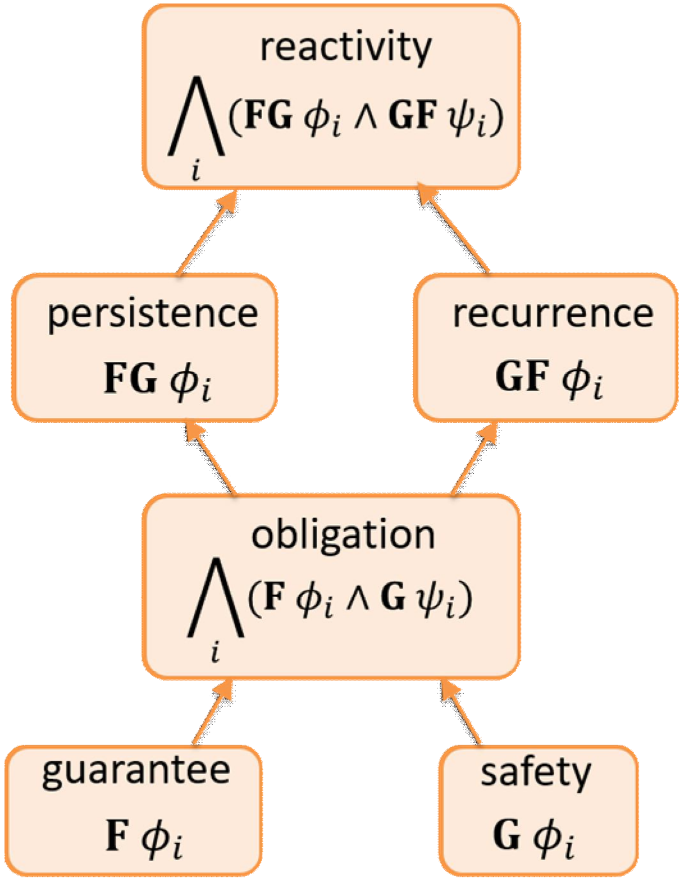
Proof rules for different classes in the hierarchy:

$$\begin{array}{ll} \text{I1.} & \Theta \rightarrow \varphi \\ \text{I2.} & \varphi \rightarrow q \\ \text{I3.} & \{\varphi\}\ \mathcal{T}\ \{\varphi\} \\ \hline & \square q \end{array}$$

$$\begin{array}{ll} \text{C1.} & \square(p \rightarrow (q \vee \varphi)) \\ \text{C2.} & \{\varphi\}\ \mathcal{T}\ \{q \vee \varphi\} \\ \text{C3.} & \{\varphi\}\ \tau\ \{q\} \\ \text{C4.} & \mathcal{T} - \{\tau\} \vdash \square(\varphi \rightarrow \diamond(q \vee En(\tau))) \\ \hline & \square(p \rightarrow \diamond q) \end{array}$$

$$\begin{array}{ll} \text{B1.} & \square(p \rightarrow (q \vee \varphi)) \\ \text{B2.} & \{\varphi \wedge (\delta = \alpha)\}\ \mathcal{T}\ \{q \vee (\varphi \wedge (\delta \preceq \alpha))\} \\ \text{B3.} & \square([\varphi \wedge (\delta = \alpha) \wedge r] \rightarrow \diamond[q \vee (\delta \prec \alpha)]) \\ \hline & \square((p \wedge \square\diamond r) \rightarrow \diamond q) \end{array}$$

The Safety-Progress Hierarchy



ϕ_i and ψ_i
are past formulas

<u>Normal form theorem</u>

Every formula is equivalent to a reactivity formula.

Proving the Normal Form Theorem

The Glory of The Past

Orna Lichtenstein
Dept. of Computer Science
Tel Aviv University
Ramat Aviv, Israel

Amir Pnueli
and
Lenore Zuck*
Dept. of Applied Mathematics
The Weizmann Institute of Science
Rehovot, 76100 Israel

„The proof […] is based on many previous results, including [Buc], [MNP], [C], [T] and [GPSS] which, when combined, yield the theorems almost immediately.“

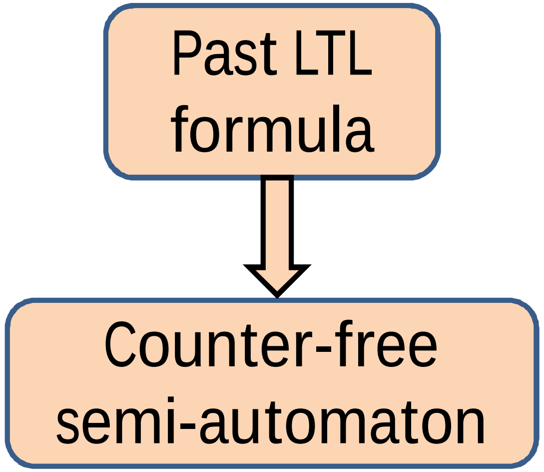
Lichtenstein, Pnueli, Zuck: Logic of Programs, 1985

Proving the Normal Form Theorem

Past LTL
formula

Zuck, PhD Thesis, 1986

Proving the Normal Form Theorem

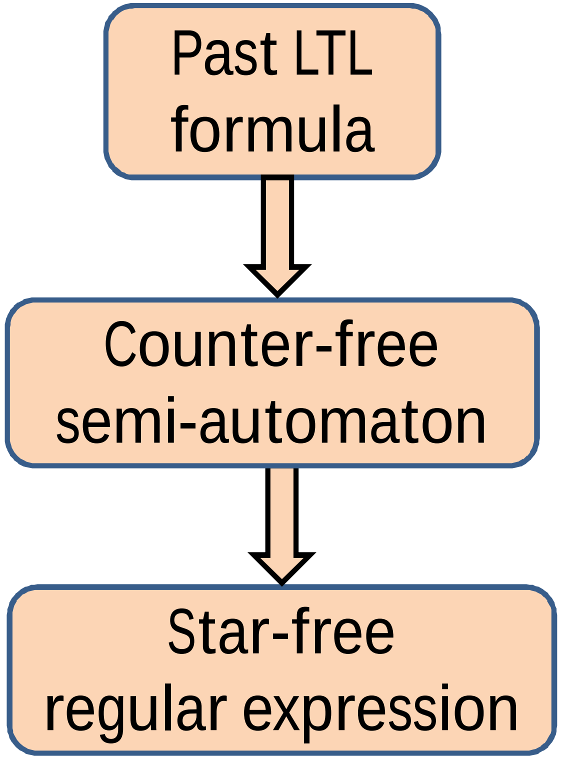


Zuck, PhD Thesis, 1986

Chapter 4

Proving the Normal Form Theorem

Zuck, PhD Thesis, 1986



Chapter 4

Krohn-Rhodes
Decomposition Theorem

Proving the Normal Form Theorem

Zuck, PhD Thesis, 1986

Past LTL formula

⇓ Chapter 4

Counter-free semi-automaton

⇓ Krohn-Rhodes Decomposition Theorem

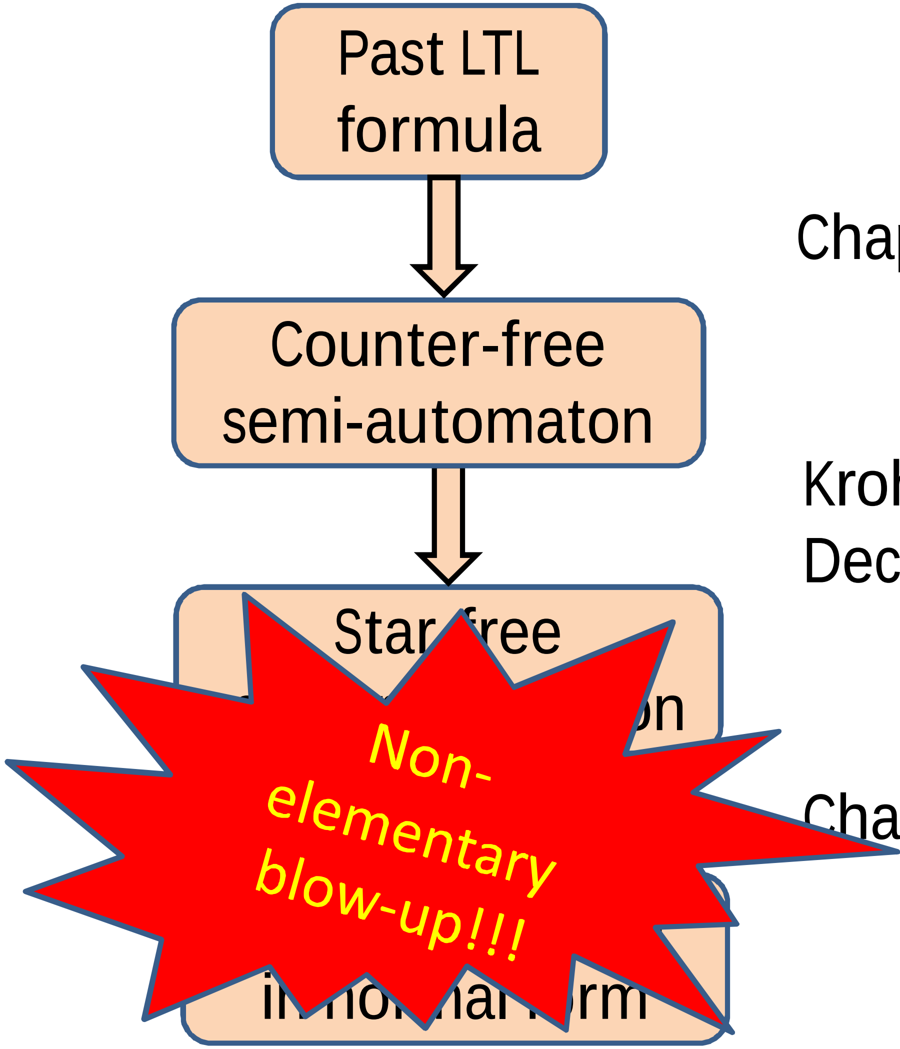
Star-free regular expression

⇓ Chapter 5

Past LTL formula in normal form

Proving the Normal Form Theorem

Zuck, PhD Thesis, 1986



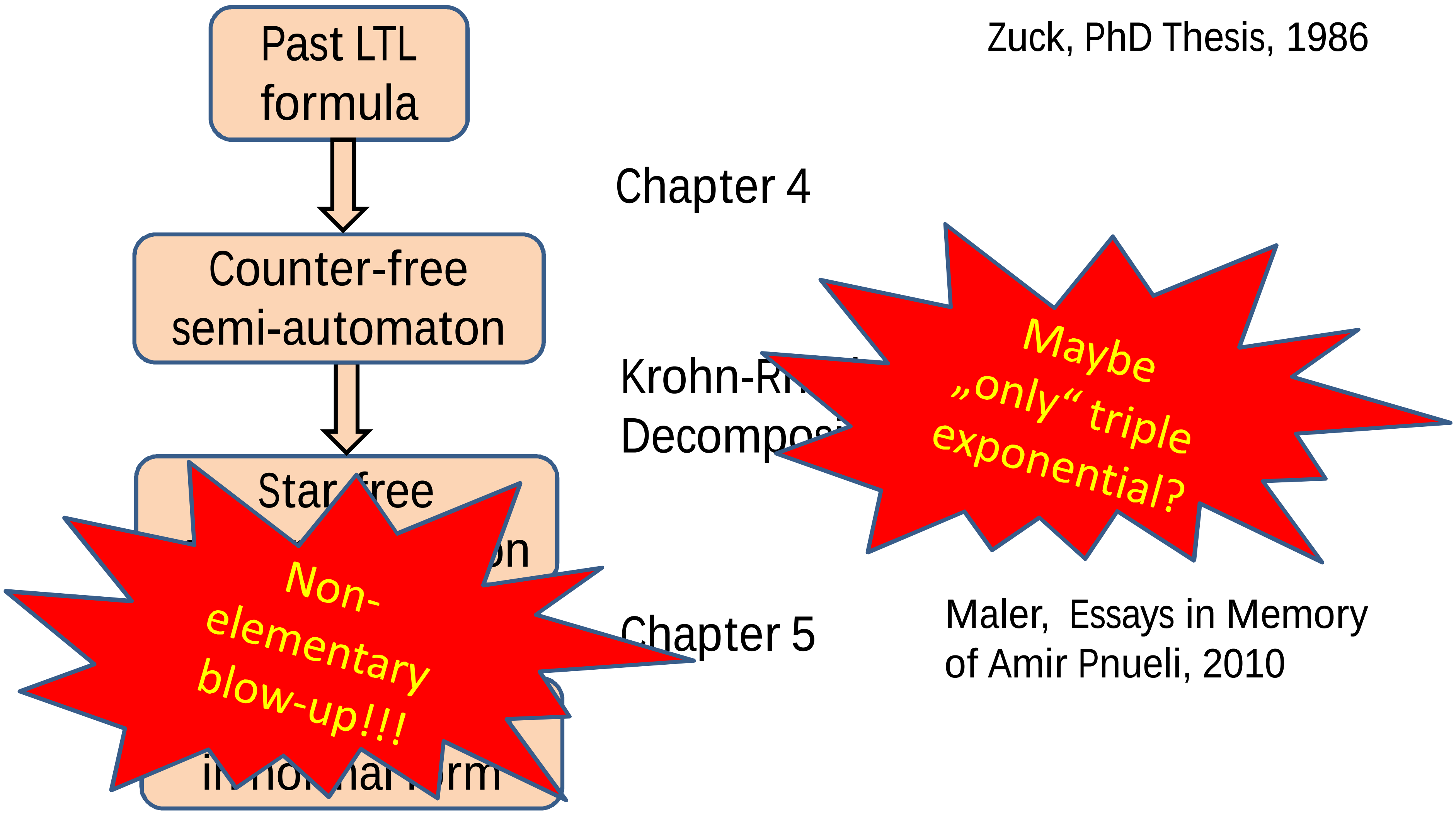
Chapter 4

Krohn-Rhodes Decomposition Theorem

Chapter 5

Proving the Normal Form Theorem

Zuck, PhD Thesis, 1986



Maler, Essays in Memory of Amir Pnueli, 2010

... and the rest is silence.

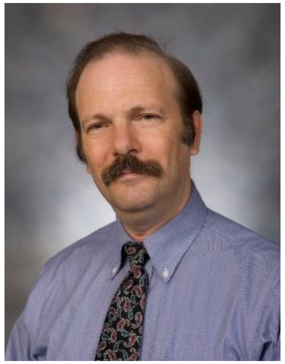
No further attempts to improve on these bounds, even though there is no lower bound!

How come?

... and the rest is silence.

No further attempts to improve on these bounds, even though there is no lower bound!

An Automata-Theoretic Approach to Automatic Program Verification



Moshe Y. Vardi

CSLI, Ventura Hall,
Stanford University,
Stanford, CA 94305.

Pierre Wolper

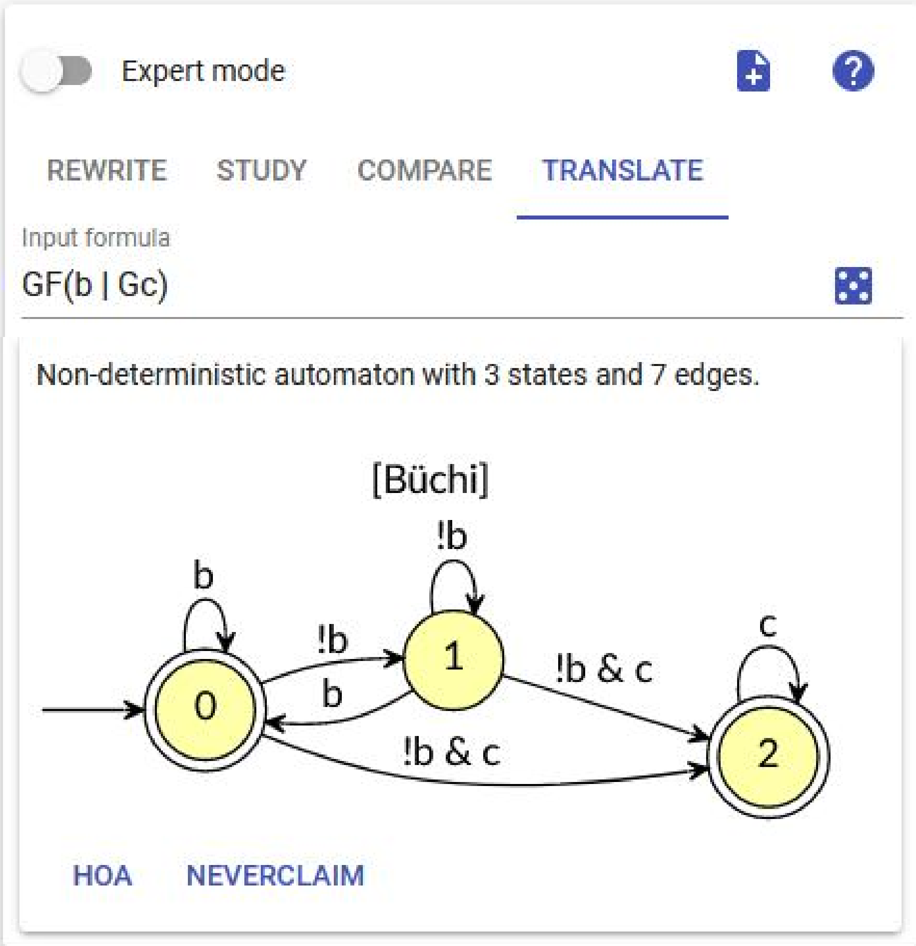
AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, NJ 07974

Gödel Prize
2000

LICS '86

Automata-theoretic approach

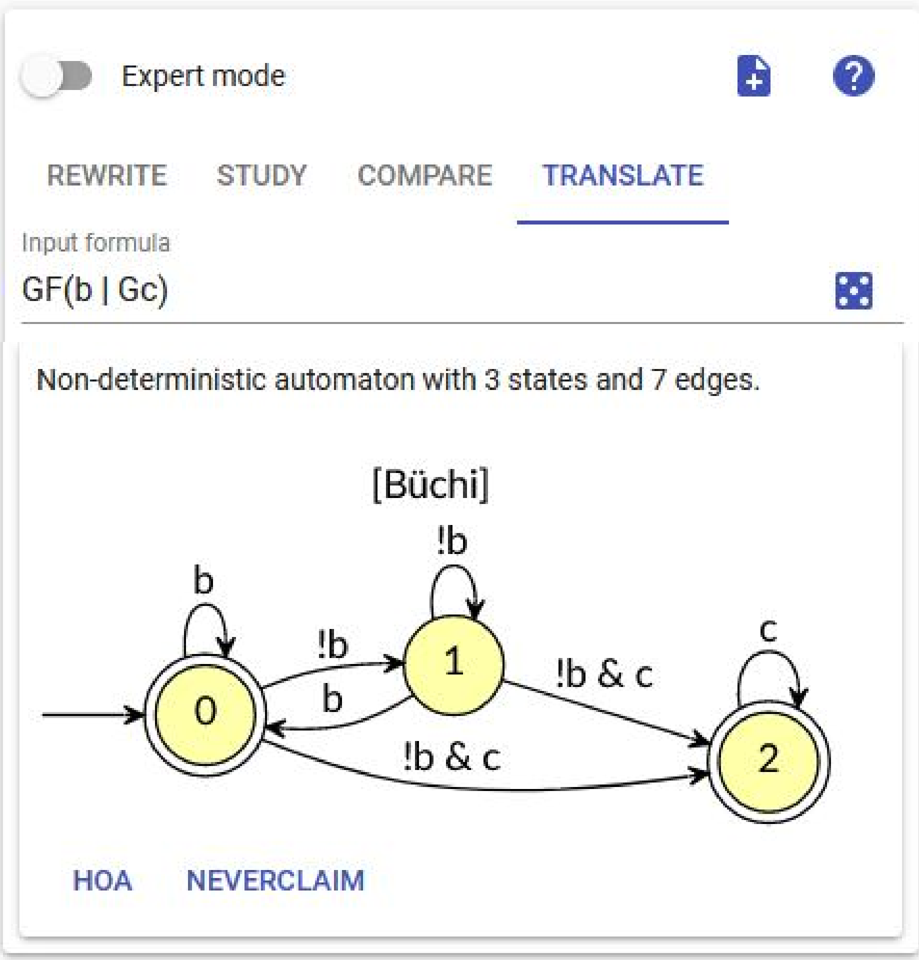
- Translates the formula into an ω-automaton (automaton on infinite words) and „throws the formula away“
- Proofs replaced by automata-theoretic algorithms
- No need for hierarchies, proof rules, or axiom systems



Duret-Lutz: Spot Online Translator
https://spot.lrde.epita.fr/app/

Automata-theoretic approach

- Translates the formula into an ω-automaton (automaton on infinite words) and „throws the formula away“
- Proofs replaced by automata-theoretic algorithms
- No need for hierarchies, proof rules, or axiom systems
- **LTL „demoted“ to syntax for automata**



Duret-Lutz: Spot Online Translator
https://spot.lrde.epita.fr/app/

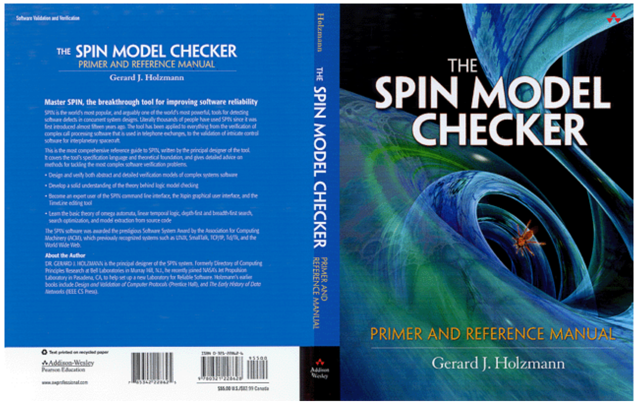
Automata-theoretic approach

During the next decades the automata-theoretic approach

Automata-theoretic approach

During the next decades the automata-theoretic approach

- is implemented in sophisticated, very successful tools

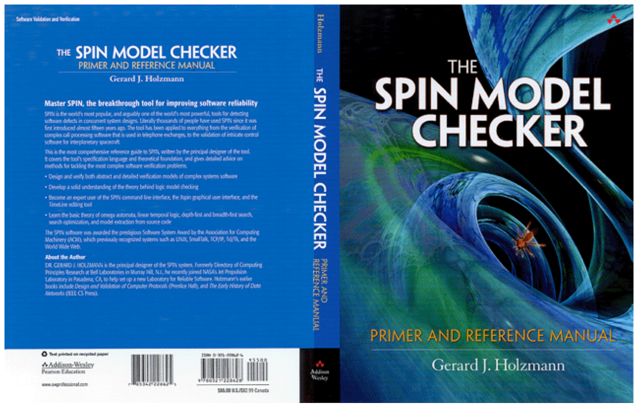


2004

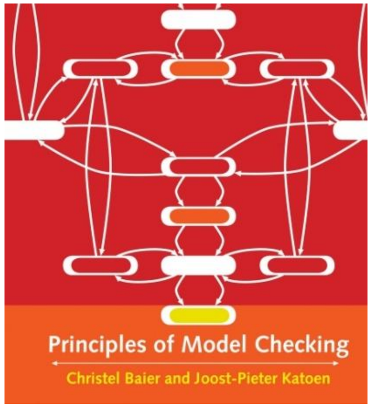
Automata-theoretic approach

During the next decades the automata-theoretic approach

- is implemented in sophisticated, very successful tools
- is extended to the verification of probabilistic systems



2004

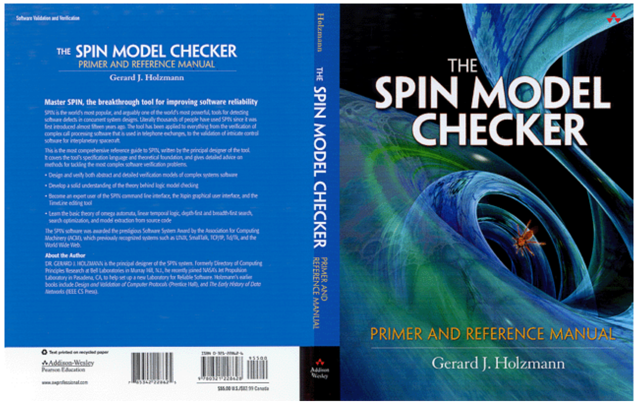


2008

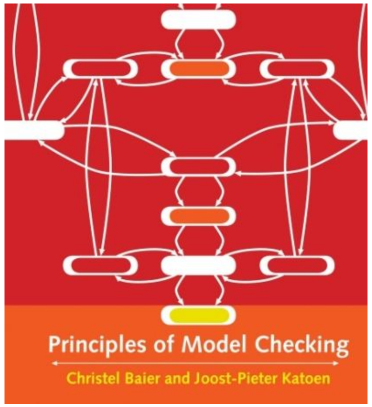
Automata-theoretic approach

During the next decades the automata-theoretic approach

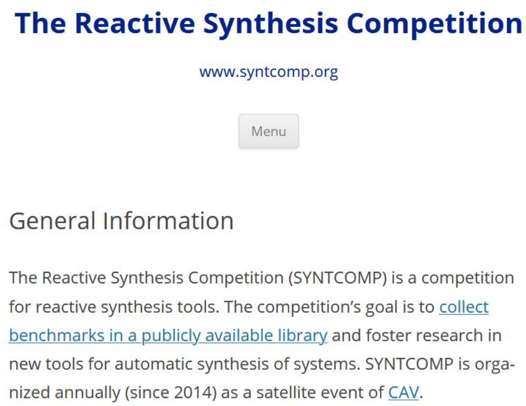
- is implemented in sophisticated, very successful tools
- is extended to the verification of probabilistic systems
- is applied to reactive synthesis: automatic synthesis of reactive systems from LTL specifications



2004



2008



2014-today

The challenge

- Reactive synthesis requires to translate LTL into deterministic ω-automata

- Probabilistic verification requires to translate LTL into limit deterministic (or deterministic) ω-automata

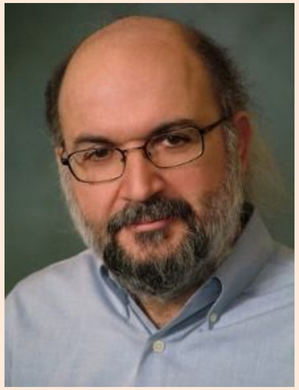
The theoretical challenge

On The Complexity of ω-Automata*

Shmuel Safra

Department of Applied Mathematics
Weizmann Institute of Science
Rehovot 76100, Israel

FOCS 1988: Determinization procedure for ω-automata



The Complexity of Probabilistic Verification

COSTAS COURCOUBETIS

University of Crete, and ICS, Farth, Heraklion, Greece

AND

MIHALIS YANNAKAKIS

AT&T Bell Laboratories, Murray Hill, New Jersey

(see also Vardi 1985)

JACM 1995: Limit-determinization procedure for ω-automata

The algorithmic challenge

These translations

- have double-exponential blow-up.
  (contrary to single-exponential for LTL $\rightarrow$ nondet. automata)
- are „monolithic“ and very combinatorial

  e.g. states of Safra‘s det. automaton are
  trees of sets of states of the nondet. automaton

  $$\Longrightarrow$$

  Implementations struggle to control
  combinatorial explosion

The algorithmic challenge

$$\bigwedge_{i=1}^{j}(\mathbf{GF}a_i) \implies \bigwedge_{i=1}^{j}(\mathbf{GF}b_i)$$

$$k\colon \bigwedge_{i=1}^{k}(\mathbf{GF}a_i \vee \mathbf{FG}b_i)$$

$$f(0,j) = (\mathbf{GF}a_0)\mathbf{U}(\mathbf{X}^j b)$$

$$f(i+1,j) = (\mathbf{GF}a_{i+1})\mathbf{U}(\mathbf{G}f(i,j))$$

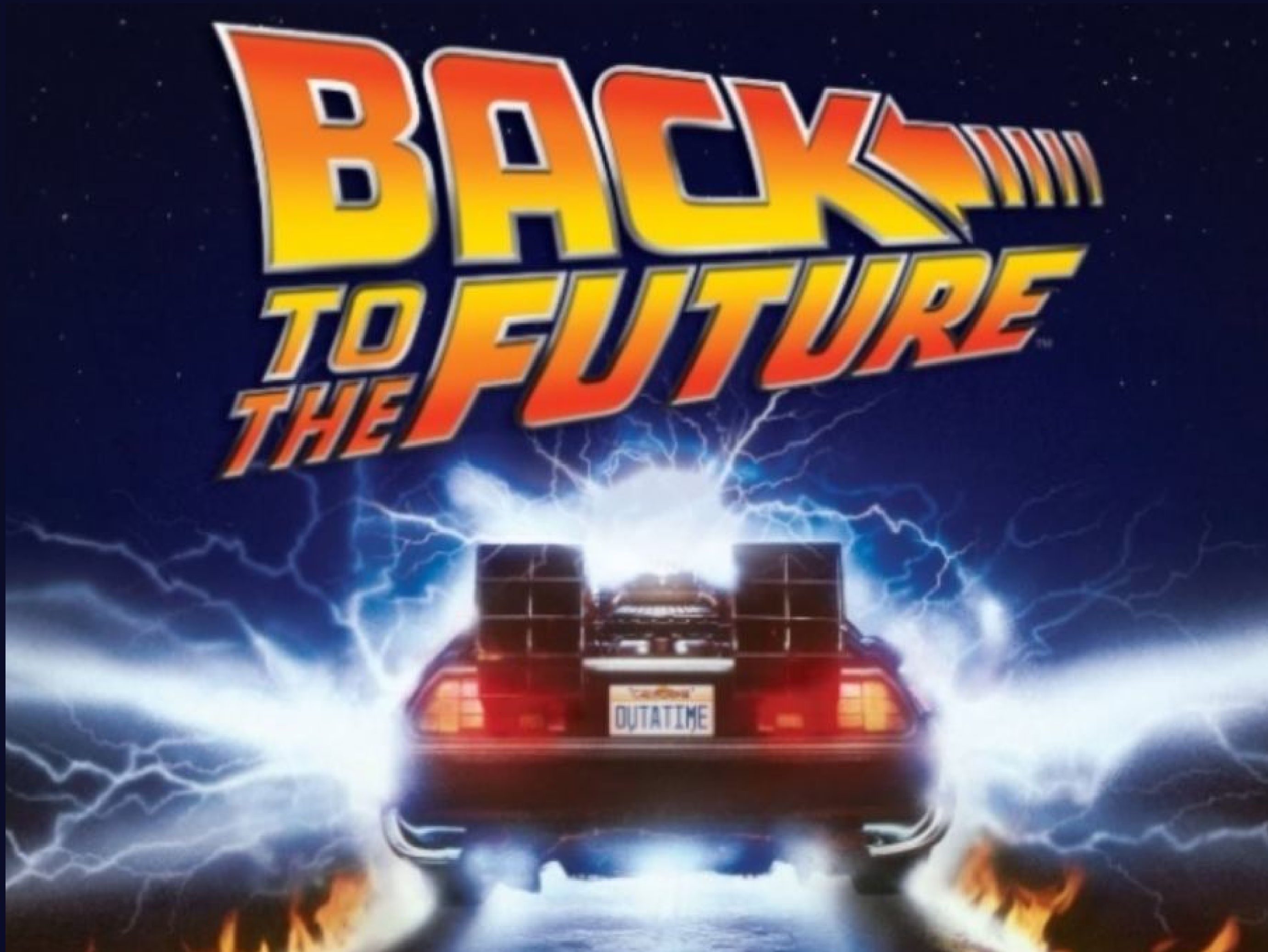
	Safra (spot+ltl2dstar)
$j = 1$	5
$j = 2$	17
$j = 3$	49
$j = 4$	129
$k = 2$	4385
$k = 3$	*
$f(0,0)$	5
$f(0,2)$	10
$f(0,4)$	12
$f(1,0)$	196
$f(1,2)$	109839
$f(1,4)$	*
$f(2,0)$	99793
$f(2,2)$	*
$f(2,4)$	*

S. Sickert, J. Esparza, S. Jaax and J. Kretinsky. CAV 2016

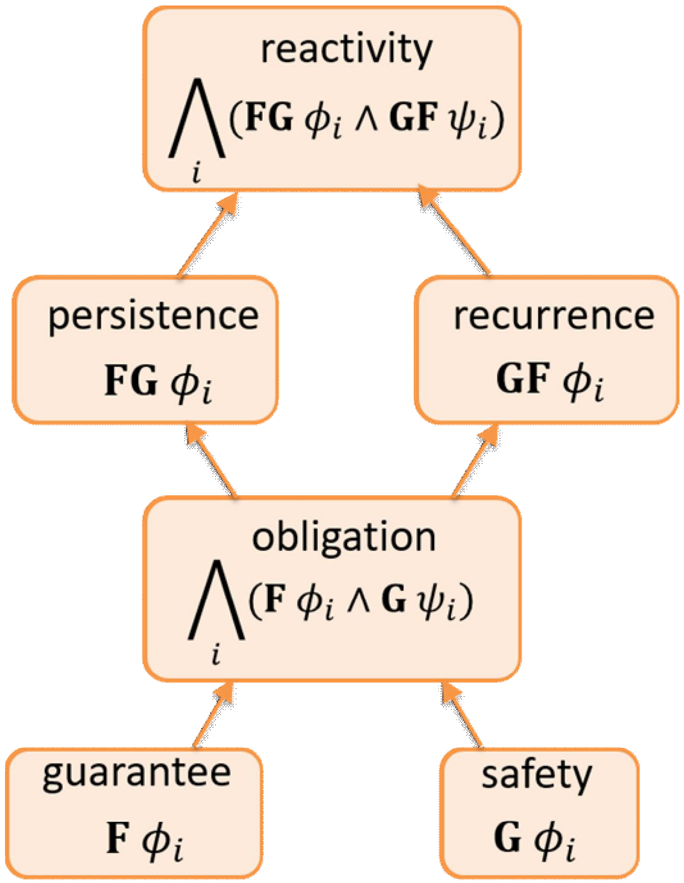
How can I do better in the future?

How can I do better in the future?

Do better in the past!


BACK TO THE FUTURE
OUTATIME

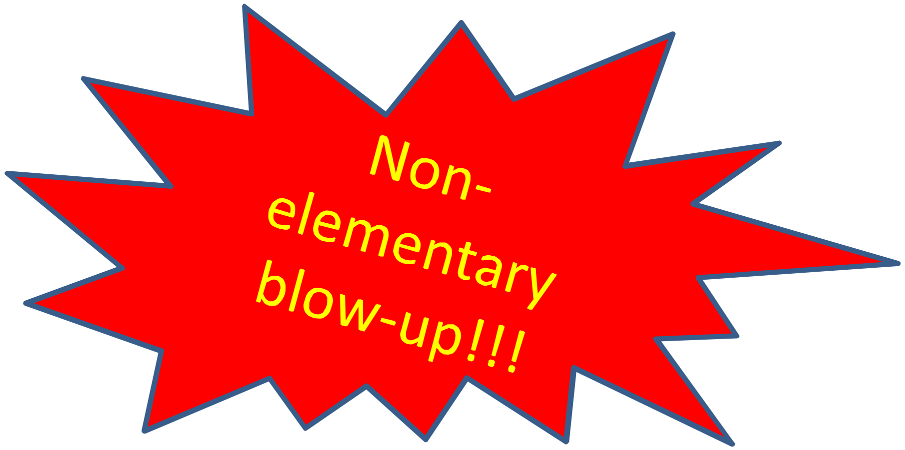
Back to the 1980s: The Safety-Progress Hierarchy



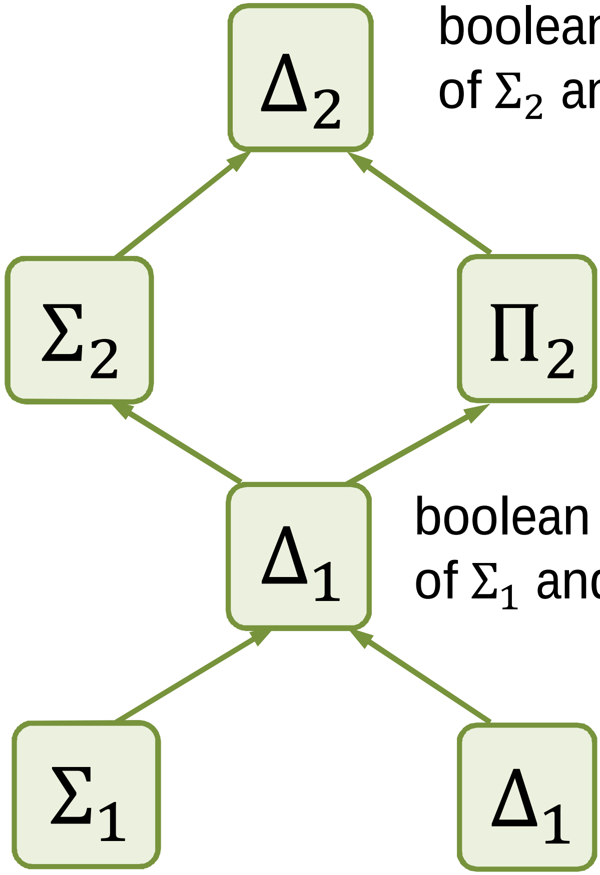
ϕ_i and ψ_i
are past formulas

<u>Normal form theorem</u>

Every formula is equivalent
to a reactivity formula.



The Alternation Hierarchy



boolean combination of Σ_2 and Π_2

at most one alternation from **U** to **W**, e.g. $\phi\ \mathbf{U}\ (\psi\ \mathbf{U}\ (\chi\ \mathbf{W}\ \rho))$

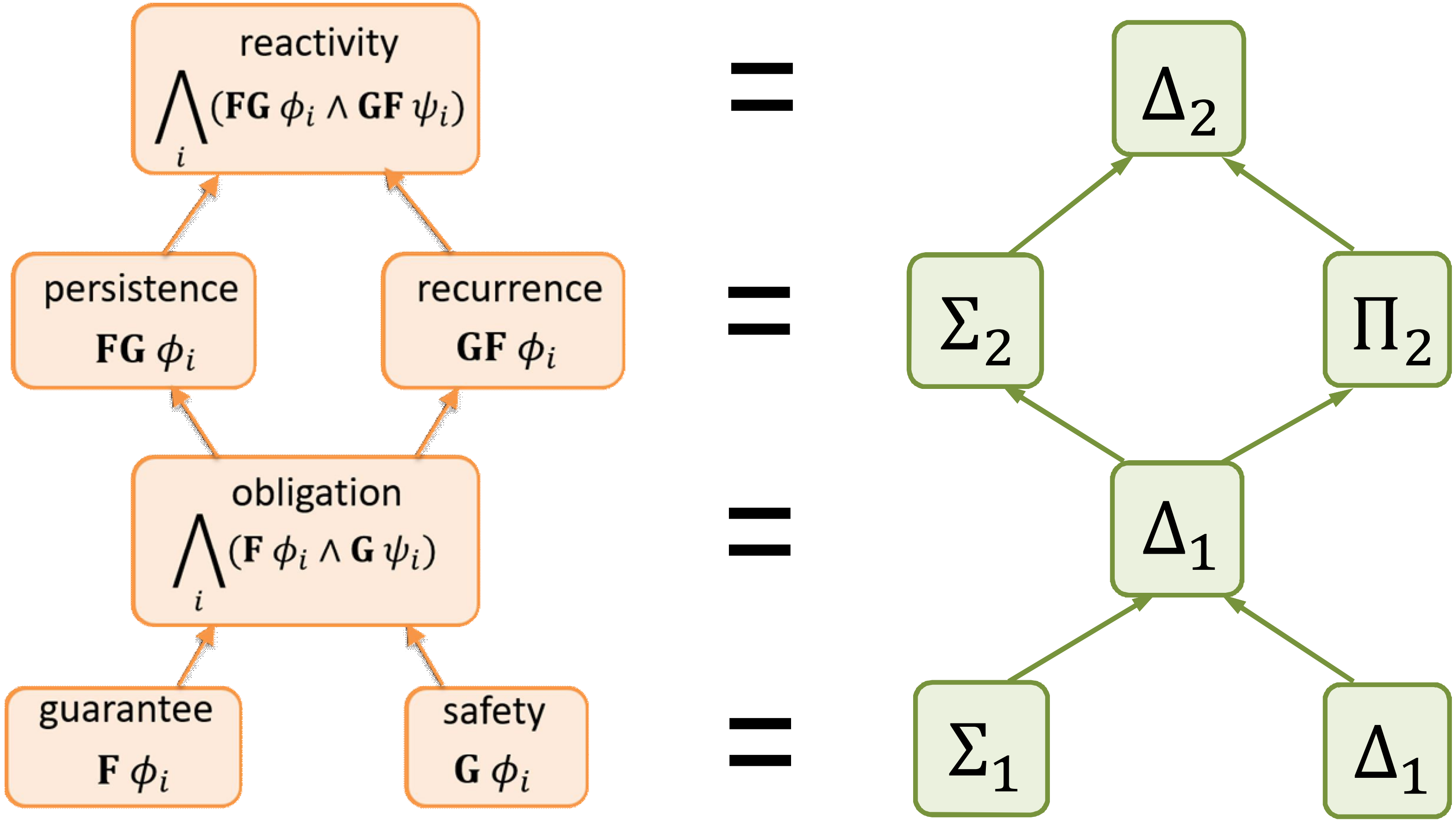
at most one alternation from **W** to **U**, e.g. $\phi\ \mathbf{W}\ (\psi\ \mathbf{U}\ (\chi\ \mathbf{U}\ \rho))$

boolean combination of Σ_1 and Π_1

only **U**, e.g. $\phi\ \mathbf{U}\ (\psi\ \mathbf{U}\ \chi)$

only **W**, e.g. $\phi\ \mathbf{W}\ (\psi\ \mathbf{W}\ \chi)$

The Alternation Hierarchy

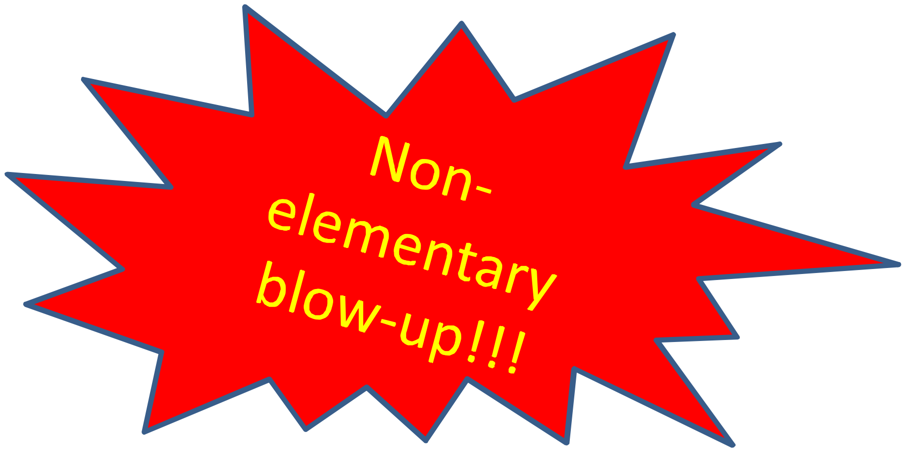


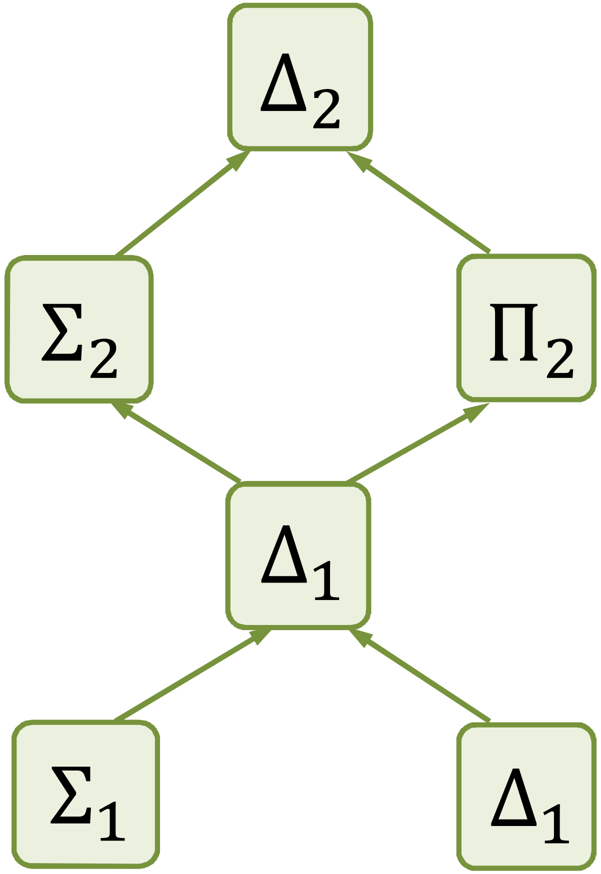
Chang, Manna, and Pnueli, ICALP 1992.
Pelánek and Strejček, CIAA 2005

The Alternation Hierarchy

<u>Normal form theorem</u>

Every formula is equivalent to a Δ_2-formula.





Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\right)$$

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee \mathbf{F}c\right)\right)$$

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\right)$$

Case 1: **$\mathbf{F}c$ holds infinitely often ($\mathbf{GF}c$ holds)**

Case 2: **$\mathbf{F}c$ only holds finitely often ($\neg\,\mathbf{GF}c$ holds)**

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\right)$$

Case 1: $\mathbf{F}c$ **holds infinitely often** ($\mathbf{GF}c$ **holds)**

Then $\mathbf{G}\,(b \vee \mathbf{F}\,c) \equiv^{\mathbf{GF}c} \mathbf{true}$

Case 2: $\mathbf{F}c$ **only holds finitely often** ($\neg\,\mathbf{GF}c$ **holds)**

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\right)$$

Case 1: $\mathbf{F}c$ holds infinitely often ($\mathbf{GF}c$ holds)

Then $\mathbf{G}\,(b \vee \mathbf{F}\,c) \equiv^{\mathbf{GF}c} \mathbf{true}$

Case 2: $\mathbf{F}c$ only holds finitely often ($\neg\mathbf{GF}c$ holds)

Then $\mathbf{G}\,(b \vee \mathbf{F}\,c) \equiv^{\neg\mathbf{GF}c} (b \vee \mathbf{F}\,c)\ \mathbf{U}\ (\mathbf{G}\,b)$

Demystifying normalization

$$\mathbf{F}\big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\big)$$

Case 1: $\mathbf{F}c$ **holds infinitely often** **($\mathbf{GF}c$ holds)**

Then $\mathbf{G}\,(b \vee \mathbf{F}\,c) \equiv^{\mathbf{GF}c} \mathbf{true}$

Case 2: $\mathbf{F}c$ **only holds finitely often** **($\neg\mathbf{GF}c$ holds)**

Then $\mathbf{G}\,(b \vee \mathbf{F}\,c) \equiv^{\neg\mathbf{GF}c} (b \vee \mathbf{F}\,c)\,\mathbf{U}\,(\mathbf{G}\,b)$

WU → UW !!

Photo by freepik –
www.freepik.com

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\right)$$

Demystifying normalization

$$\mathbf{F}\Big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\Big) \;\equiv\; \begin{array}{c} \mathbf{GF}c \wedge \mathbf{F}\Big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\Big) \\ \vee \\ \neg\,\mathbf{GF}c \wedge \mathbf{F}\Big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\Big) \end{array}$$

Demystifying normalization

$$\mathbf{F}\big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\big) \;\equiv\; \mathbf{GF}c \wedge \mathbf{F}\big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\big)$$

$$\vee$$

$$\neg\,\mathbf{GF}c \wedge \mathbf{F}\big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\big)$$

Demystifying normalization

$$\mathbf{F}\Big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\Big) \;\equiv\; \mathbf{GF}c \wedge \mathbf{F}(a \wedge \mathbf{true})$$

$$\vee$$

$$\neg\,\mathbf{GF}c \wedge \mathbf{F}\Big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\Big)$$

Demystifying normalization

$$\mathbf{F}\big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\big) \;\equiv\; \mathbf{GF}c \wedge \mathbf{F}(a \wedge \mathbf{true})$$

$$\vee$$

$$\neg\,\mathbf{GF}c \wedge \mathbf{F}\big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\big)$$

Correct because $\mathbf{GF}c$
holds at some moment
iff it holds at every moment!

Demystifying normalization

$$\mathbf{F}\big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\big) \equiv \mathbf{GF}c \wedge \mathbf{F}(a \wedge \mathbf{true})$$

$$\vee$$

$$\neg\mathbf{GF}c \wedge \mathbf{F}(a \wedge (b \vee \mathbf{F}c)\mathbf{U}\,\mathbf{G}b)$$

Correct because $\neg\mathbf{GF}c$
holds at some moment
iff it holds at every moment!

... et voilá!

$$\mathbf{F}\Big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\Big) \;\equiv\; \begin{array}{c} \mathbf{GF}c \wedge \mathbf{F}a \\ \vee \\ \mathbf{F}(a \wedge (b \vee \mathbf{F}c)\,\mathbf{U}\;\mathbf{G}b) \end{array}$$

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee \mathbf{F}(c \wedge \mathbf{G}d)\right)\right)$$

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee \mathbf{F}(c \wedge \mathbf{G}d)\right)\right)$$

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee \mathbf{F}(c \wedge \mathbf{G}d)\right)\right)$$

$$\equiv \quad \mathbf{FG}d \;\wedge\; \mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee \mathbf{F}(c \wedge \mathbf{G}d)\right)\right)$$

$$\vee$$

$$\neg\mathbf{FG}d \;\wedge\; \mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee \mathbf{F}(c \wedge \mathbf{G}d)\right)\right)$$

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee \mathbf{F}(c \wedge \mathbf{G}d)\right)\right)$$

$$\equiv \quad \mathbf{FG}d \;\wedge\; \mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee (\mathbf{F}c \;\mathbf{W}\; (c \wedge \mathbf{G}d))\right)\right)$$

$$\vee$$

$$\neg\mathbf{FG}d \;\wedge\; \mathbf{F}\left(a \wedge \mathbf{G}\,(b \vee \mathbf{false})\right)$$

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee \mathbf{F}(c \wedge \mathbf{G}d)\right)\right)$$

$$\equiv \quad \mathbf{FG}d \ \wedge \ \mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee (\mathbf{F}c\ \mathbf{W}\ \mathbf{(}c \wedge \mathbf{G}d\mathbf{)})\right)\right)$$

$$\vee$$

$$\mathbf{GF}\neg d \ \wedge \ \mathbf{F}(a \wedge \mathbf{G}b)$$

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee \mathbf{F}(c \wedge \mathbf{G}d)\right)\right)$$

$$\equiv \quad \mathbf{FG}d \;\wedge\; \boxed{\mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee (\mathbf{F}c\ \mathbf{W}\ (c \wedge \mathbf{G}d))\right)\right)}$$

$$\vee$$

$$\mathbf{GF}\neg d \;\wedge\; \mathbf{F}(a \wedge \mathbf{G}b)$$

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee \mathbf{F}(c \wedge \mathbf{G}d)\right)\right)$$

$$\equiv \quad \mathbf{FG}d \;\wedge\; \mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee (\mathbf{F}c\ \mathbf{W}\ (c \wedge \mathbf{G}d))\right)\right)$$

$$\vee$$

$$\mathbf{GF}\neg d \;\wedge\; \mathbf{F}(a \wedge \mathbf{G}b)$$

Demystifying normalization

$$\mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee \mathbf{F}(c \wedge \mathbf{G}d)\right)\right)$$

$$\equiv \quad \mathbf{FG}d \;\wedge \left(\begin{array}{c} \mathbf{GF}c \wedge \mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee (\mathbf{F}c\ \mathbf{W}\ (c \wedge \mathbf{G}d))\right)\right) \\ \vee \\ \neg\mathbf{GF}c \wedge \mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee (\mathbf{F}c\ \mathbf{W}\ (c \wedge \mathbf{G}d))\right)\right) \end{array} \right)$$

$$\vee$$

$$\mathbf{GF}\neg d \;\wedge\; \mathbf{F}(a \wedge \mathbf{G}b)$$

Demystifying normalization

$$\mathbf{F}\Big(a \wedge \mathbf{G}\big(b \vee \mathbf{F}(c \wedge \mathbf{G}d)\big)\Big)$$

$$\equiv \quad \mathbf{FG}d \;\wedge \left(\begin{array}{c} \mathbf{GF}c \wedge \mathbf{F}(a \wedge \mathbf{true}) \\ \vee \\ \mathbf{FG}\neg c \wedge \mathbf{F}\Big(a \wedge \mathbf{F}(c \wedge \mathbf{G}d)\ \mathbf{U}\ \mathbf{G}\big(b \vee (c \wedge \mathbf{G}d)\big)\Big) \end{array} \right)$$

$$\vee$$

$$\mathbf{GF}\neg d \;\wedge\; \mathbf{F}(a \wedge \mathbf{G}b)$$

... et voilá!

$$\mathbf{F}\left(a \wedge \mathbf{G}\left(b \vee \mathbf{F}(c \wedge \mathbf{G}d)\right)\right)$$

$$\equiv \quad \mathbf{FG}d \wedge \mathbf{GF}c \wedge \mathbf{F}a$$

$$\vee$$

$$\mathbf{FG}d \wedge \mathbf{F}\left(a \wedge \mathbf{F}(c \wedge \mathbf{G}d)\ \mathbf{U}\ \mathbf{G}\left(b \vee (c \wedge \mathbf{G}d)\right)\right)$$

$$\vee$$

$$\mathbf{F}(a \wedge \mathbf{G}b)$$

Closed-form expression

$$\varphi \equiv \bigvee_{\substack{M \subseteq \mathbf{U}(\varphi) \\ N \subseteq \mathbf{W}(\varphi)}} \left(\varphi' \wedge \bigwedge_{\psi \in M} \mathbf{GF}\, \psi' \wedge \bigwedge_{\chi \in N} \mathbf{FG}\, \chi' \right)$$

Sickert and Esparza, LICS 2020
Esparza, Křetínský, and Sickert, JACM 2020

Closed-form expression

$O(2^n)$ disjuncts

$O(n)$ conjuncts of length $O(n)$

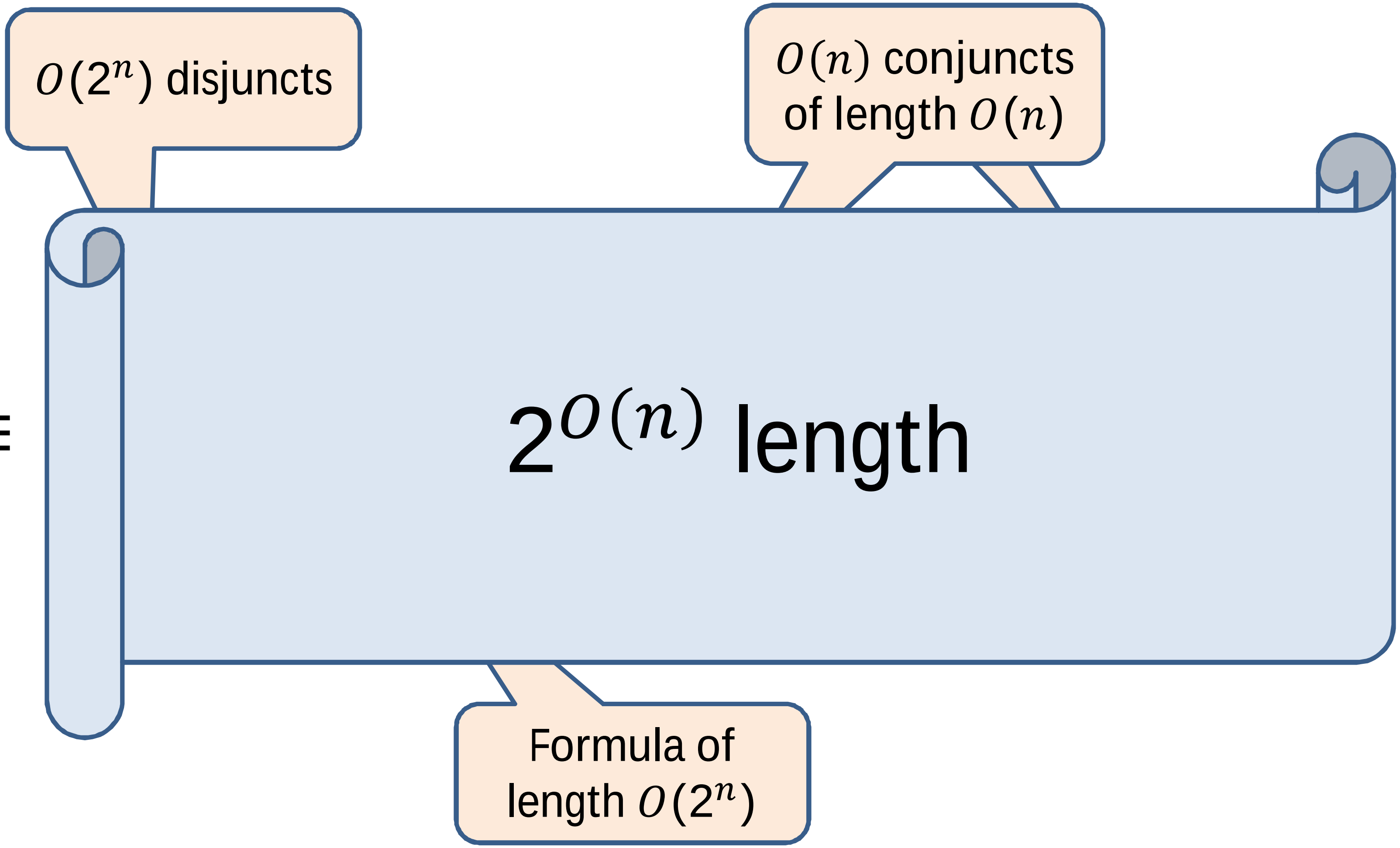
$$\varphi \equiv \bigvee_{\substack{M \subseteq \mathbf{U}(\varphi) \\ N \subseteq \mathbf{W}(\varphi)}} \left(\varphi' \wedge \bigwedge_{\psi \in M} \mathbf{GF}\, \psi' \wedge \bigwedge_{\chi \in N} \mathbf{FG}\, \chi' \right)$$

Formula of length $O(2^n)$

Sickert and Esparza, LICS 2020
Esparza, Křetínský, and Sickert, JACM 2020

Closed-form expression

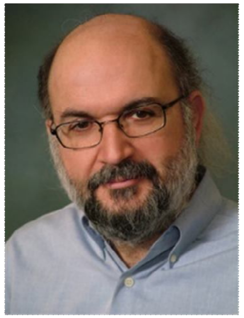
$\varphi \equiv$



Sickert and Esparza, LICS 2020
Esparza, Křetínský, and Sickert, JACM 2020

Back from the past

LTL ⇒ Limit-deterministic Büchi automata



The Complexity of Probabilistic Verification

COSTAS COURCOUBETIS

University of Crete, and ICS, Farth, Heraklion, Greece

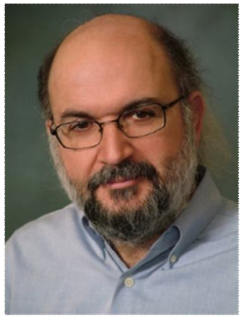
AND

MIHALIS YANNAKAKIS

AT&T Bell Laboratories, Murray Hill, New Jersey

1995: Limit-determinization procedure for ω-automata

LTL ⇒ Limit-deterministic Büchi automata



The Complexity of Probabilistic Verification

COSTAS COURCOUBETIS

University of Crete, and ICS, Farth, Heraklion, Greece

AND

MIHALIS YANNAKAKIS

AT&T Bell Laboratories, Murray Hill, New Jersey

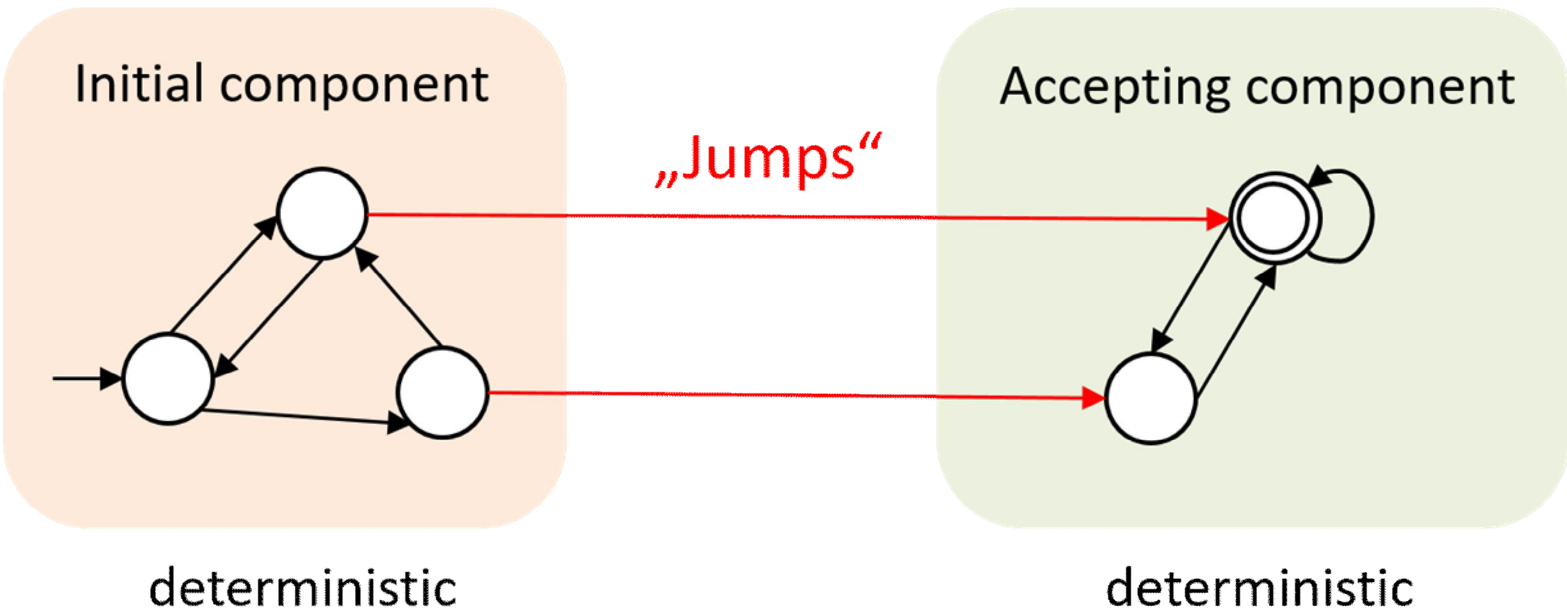
1995: Limit-determinization procedure for ω-automata

Formula

$\Longrightarrow$ Δ_2-formula

$\Longrightarrow$ Limit-deterministic Büchi automaton

LTL ⇒ Limit-deterministic Büchi automata



LTL ⇒ Limit-deterministic Büchi automata

$$\mathbf{F}\big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\big)$$

LTL ⇒ Limit-deterministic Büchi automata

$$\mathbf{F}\big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\big)$$

$$\equiv$$

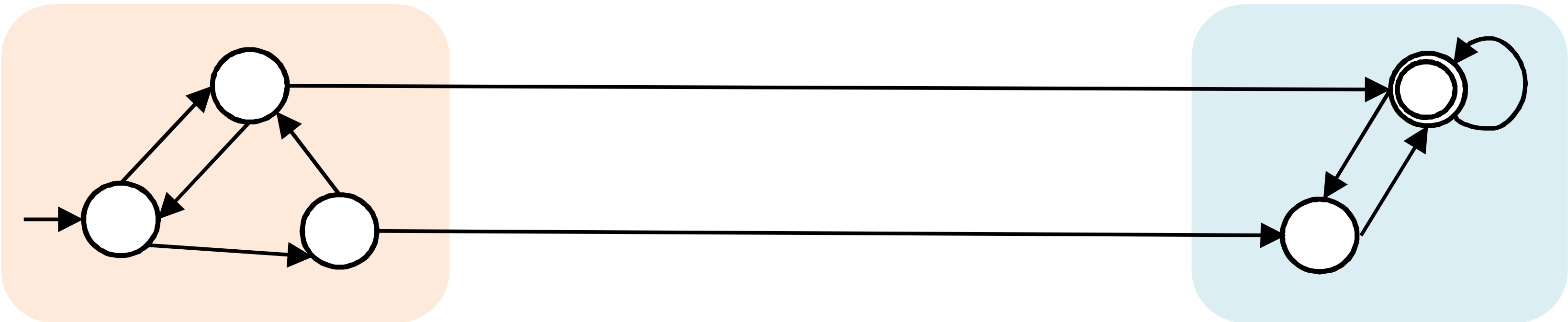
$$(\mathbf{GF}c \wedge \mathbf{F}a) \quad \vee \quad \mathbf{F}(a \wedge (b \vee \mathbf{F}c)\ \mathbf{U}\ \mathbf{G}b)$$

LTL ⇒ Limit-deterministic Büchi automata

$$\mathbf{F}\big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\big)$$

$$\equiv$$

$$(\mathbf{GF}c \wedge \mathbf{F}a) \quad \vee \quad \color{red}{\mathbf{F}(a \wedge (b \vee \mathbf{F}c)\ \mathbf{U}}\ \color{blue}{\mathbf{G}b}\color{red}{)}$$



LTL ⇒ Limit-deterministic Büchi automata

$$\mathbf{F}\big(a \wedge \mathbf{G}\,(b \vee \mathbf{F}c)\big)$$

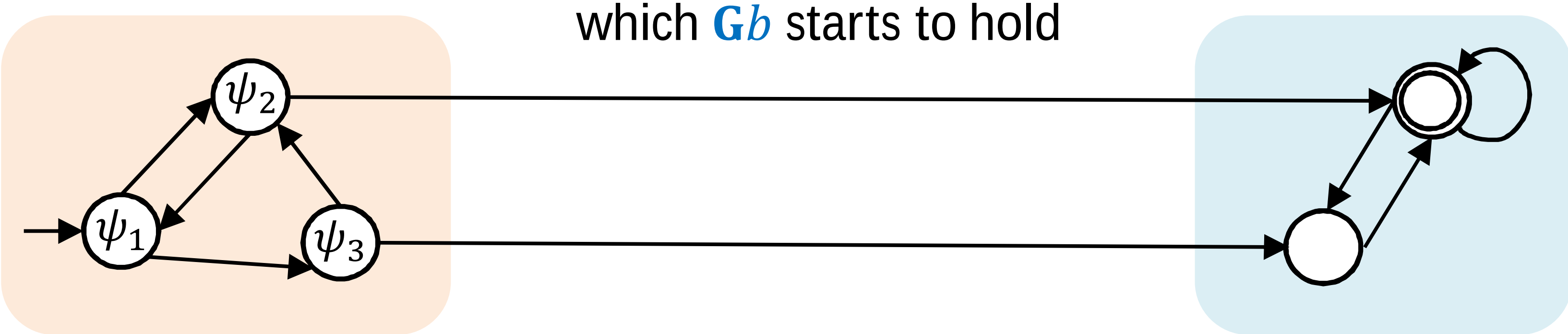
$$\equiv$$

$$(\mathbf{GF}c \wedge \mathbf{F}a) \quad \vee \quad \mathbf{F}(a \wedge (b \vee \mathbf{F}c)\ \mathbf{U}\ \mathbf{G}b)$$

Maintains the formula ψ_i that must hold when $\mathbf{G}b$ starts to hold

Guesses the point at which $\mathbf{G}b$ starts to hold

Checks $\psi_i \wedge \mathbf{G}b$



Size reduction

$$\bigwedge_{i=1}^{j}(\mathbf{GF}a_i) \implies \bigwedge_{i=1}^{j}(\mathbf{GF}b_i)$$

$$k\colon \bigwedge_{i=1}^{k}(\mathbf{GF}a_i \vee \mathbf{FG}b_i)$$

$$f(0,j) = (\mathbf{GF}a_0)\mathbf{U}(\mathbf{X}^j b)$$

$$f(i+1,j) = (\mathbf{GF}a_{i+1})\mathbf{U}(\mathbf{G}f(i,j))$$

	LDBA	Safra (spot+ltl2dstar)
$j=1$	3	5
$j=2$	4	17
$j=3$	5	49
$j=4$	6	129
$k=2$	5	4385
$k=3$	9	*
$f(0,0)$	5	5
$f(0,2)$	10	10
$f(0,4)$	16	12
$f(1,0)$	6	196
$f(1,2)$	28	109839
$f(1,4)$	58	*
$f(2,0)$	10	99793
$f(2,2)$	46	*
$f(2,4)$	92	*

Sickert, Esparza, Jaax, and Kretinsky, CAV 2016

LTL ⇒ deterministic Rabin automata

On The Complexity of ω-Automata*

Shmuel Safra

Department of Applied Mathematics
Weizmann Institute of Science
Rehovot 76100, Israel

1988: Determinization procedure for ω-automata

LTL ⇒ deterministic Rabin automata

On The Complexity of ω-Automata*

Shmuel Safra

Department of Applied Mathematics
Weizmann Institute of Science
Rehovot 76100, Israel

1988: Determinization procedure for ω-automata

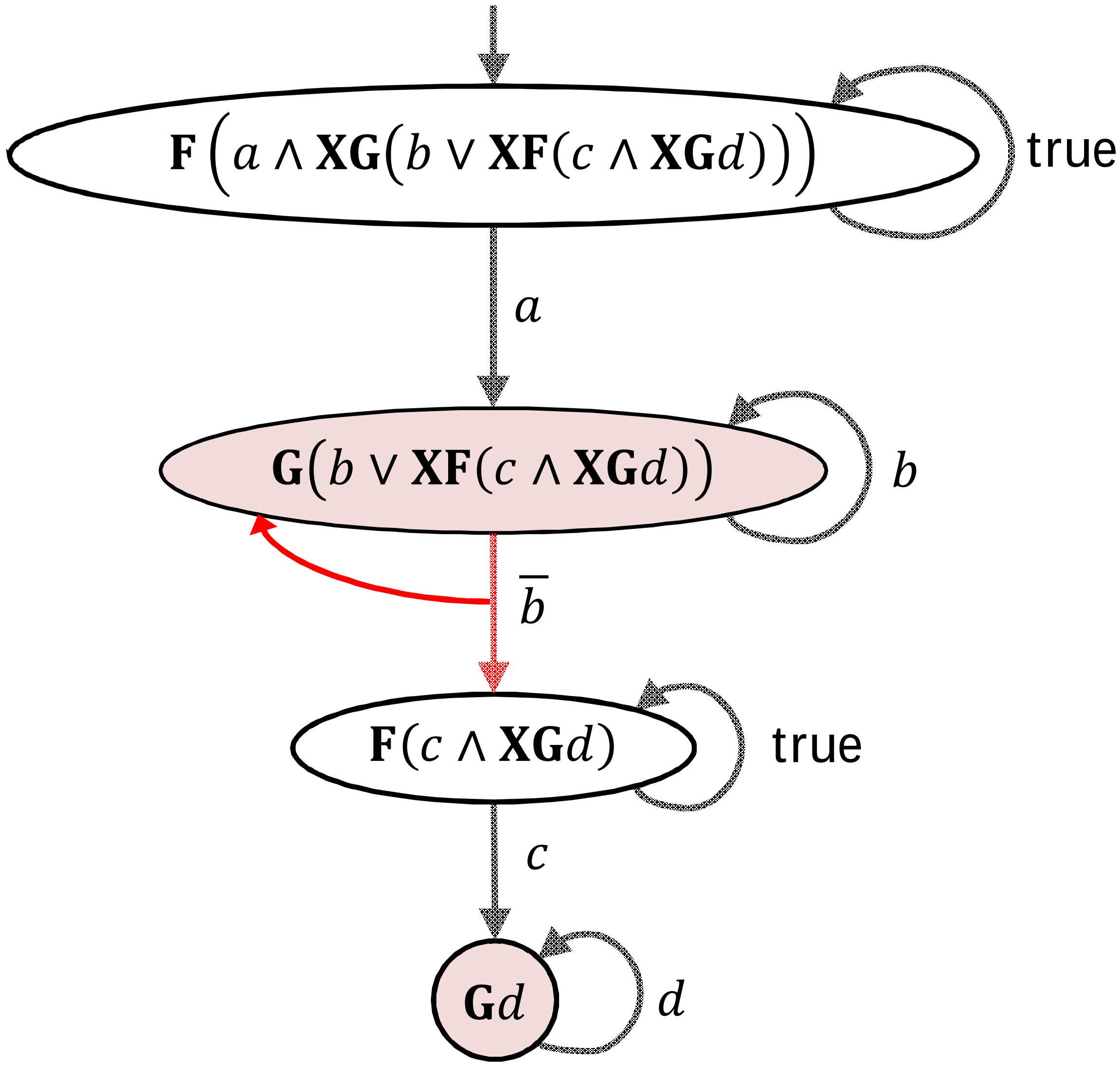
Formula

$\Longrightarrow$ Δ_2-formula

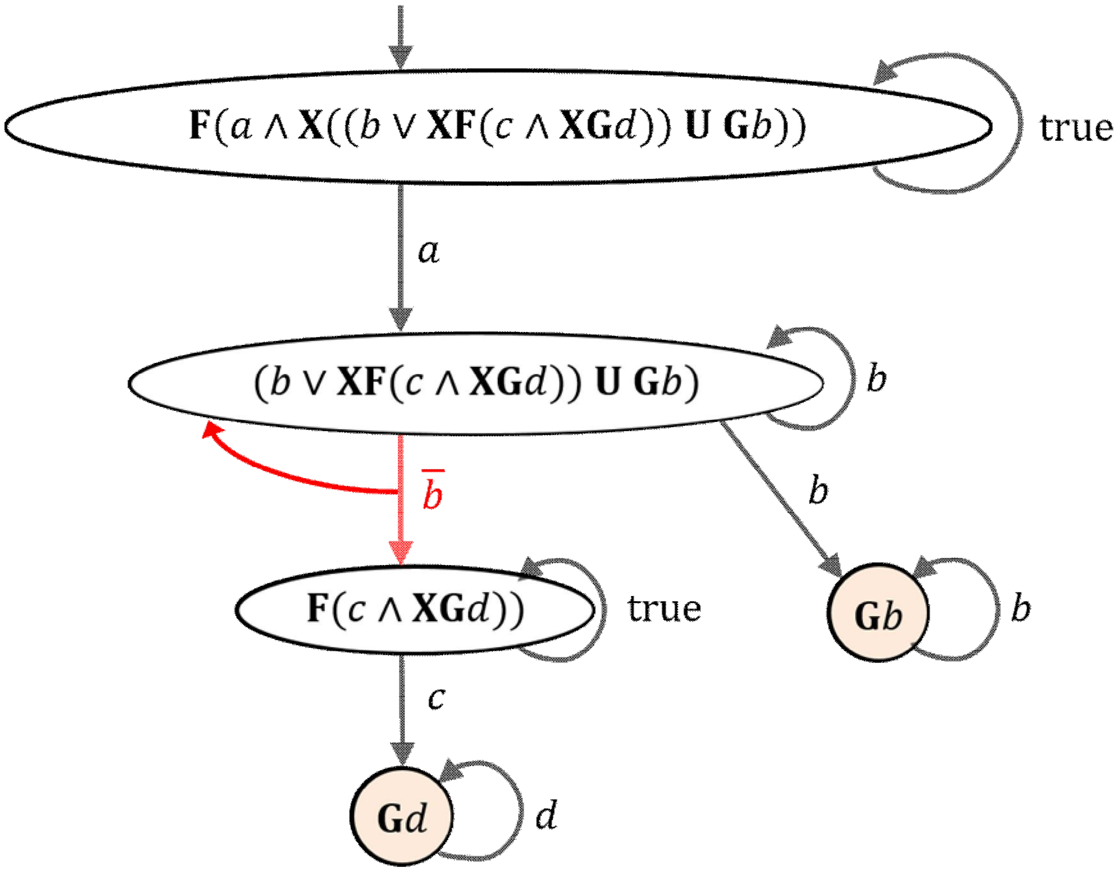
$\Longrightarrow$ very weak Δ_2-alternating Büchi automaton

$\Longrightarrow$ deterministic Rabin automaton

From LTL to very weak alternating Büchi automata

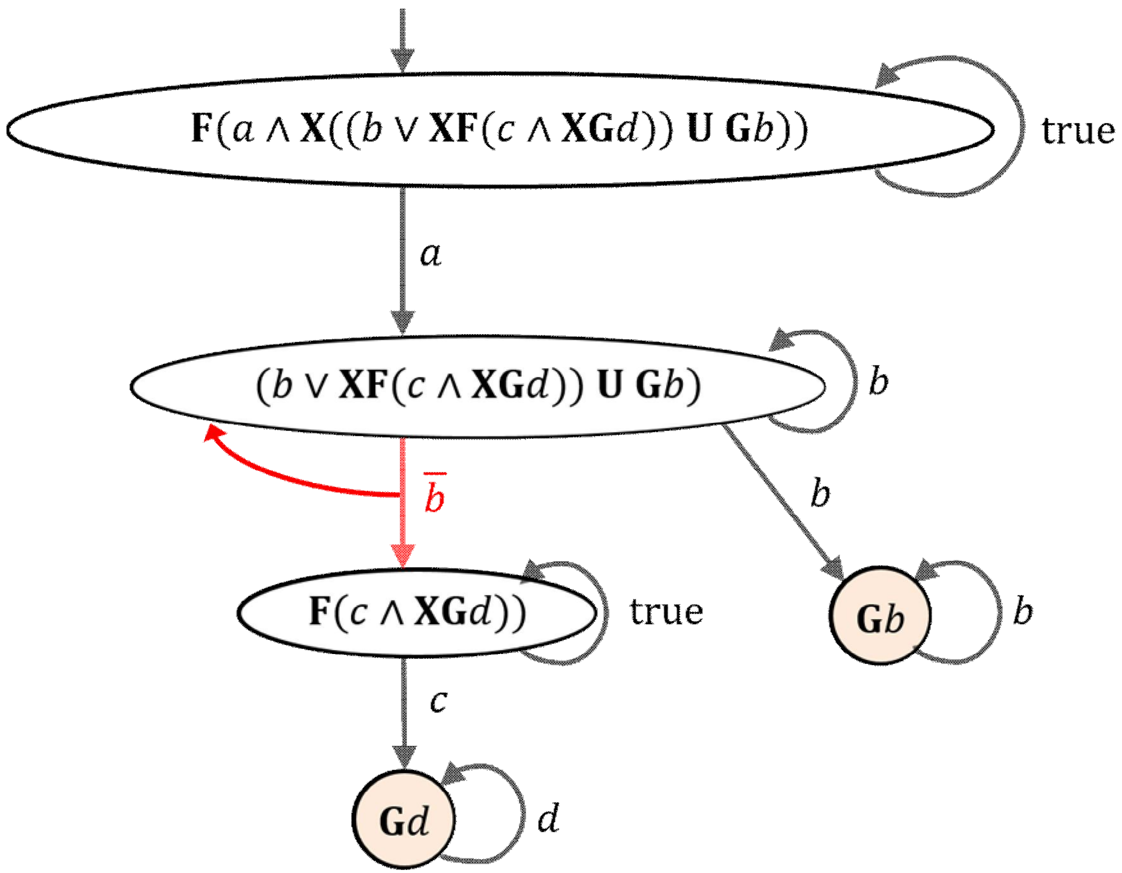


After Δ_2-normalization



Disjunction of VWAA with $O(n)$ states s.t. each path has only one alternation between accepting and non-accepting states

After Δ_2-normalization



Lemma: A Δ_2-VWAA accepts a word iff it has a run on it such that

- No level of the tree is (equiv. to) false, and
- All states of some level are accepting.

Equivalent deterministic Büchi or co-Büchi automaton using (a slight reformulation of) the breakpoint construction.

From trees of sets to pairs of sets.

Owl (owl.model.in.tum.de)

Owl

A Java tool collection and library for **O**mega-**w**ords, ω-automata and **L**inear Temporal Logic (LTL). Batteries included.

Online **Demo**	Download **ZIP File**	View On **GitLab**

Owl is a Java 11 tool collection and library for ω-words, ω-automata and linear temporal logic. It provides a wide range of algorithms for automata and LTL. It has

Křetínský , Meggendorfer, Sickert, ATVA 2018

Strix (strix.model.in.tum.de)

Tool for reactive LTL synthesis
- direct translation LTL-to-DPA
- multi-threaded, explicit-state solver for parity games.

Winner of the SYNTCOMP competition in 2018,2019, 2020

State-of-the-art in reactive LTL synthesis

Luttenberger, Meyer, Sickert,
CAV 2018 and Acta Informatica 2020

Assumptions:

```
1 true
```

Guarantees:

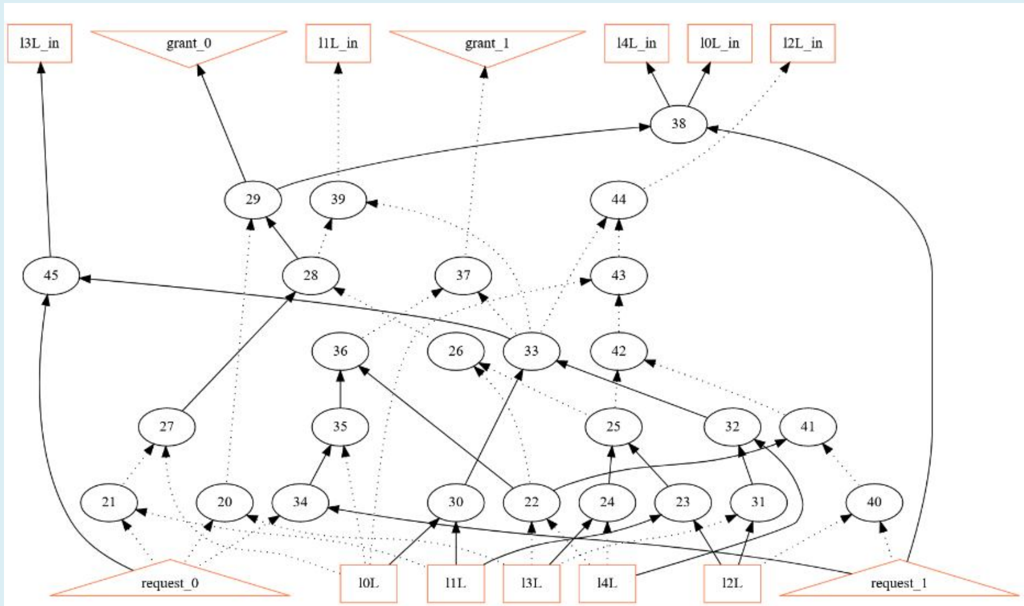
```
1 G (request_0 -> F grant_0)
2 G (request_1 -> F grant_1)
3 G (!grant_0 || !grant_1)
4 (!grant_0 & !request_0) W request_0
5 (!grant_1 & !request_1) W request_1
```

Input propositions:

```
request_0, request_1
```

Output propositions:

```
grant_0, grant_1
```



A fresh look at LTL

A fresh look at LTL

Imagine SAT without CNF

A fresh look at LTL

Imagine SAT without CNF

Imagine FOL without skolemization

A fresh look at LTL

Imagine SAT without CNF

Imagine FOL without skolemization

That‘s what happened to LTL

Thank you for your attention!

Thank you for your attention!