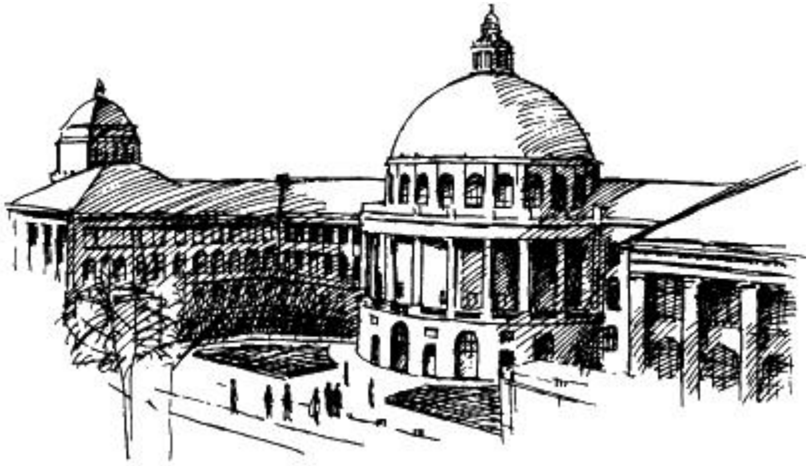
From Temporal Logics to Automata via Alternation Elimination

Christian Dax



defense talk, 2010

Scope & Motivation

NOKIA
Connecting People

BOSCH

AMD

intel

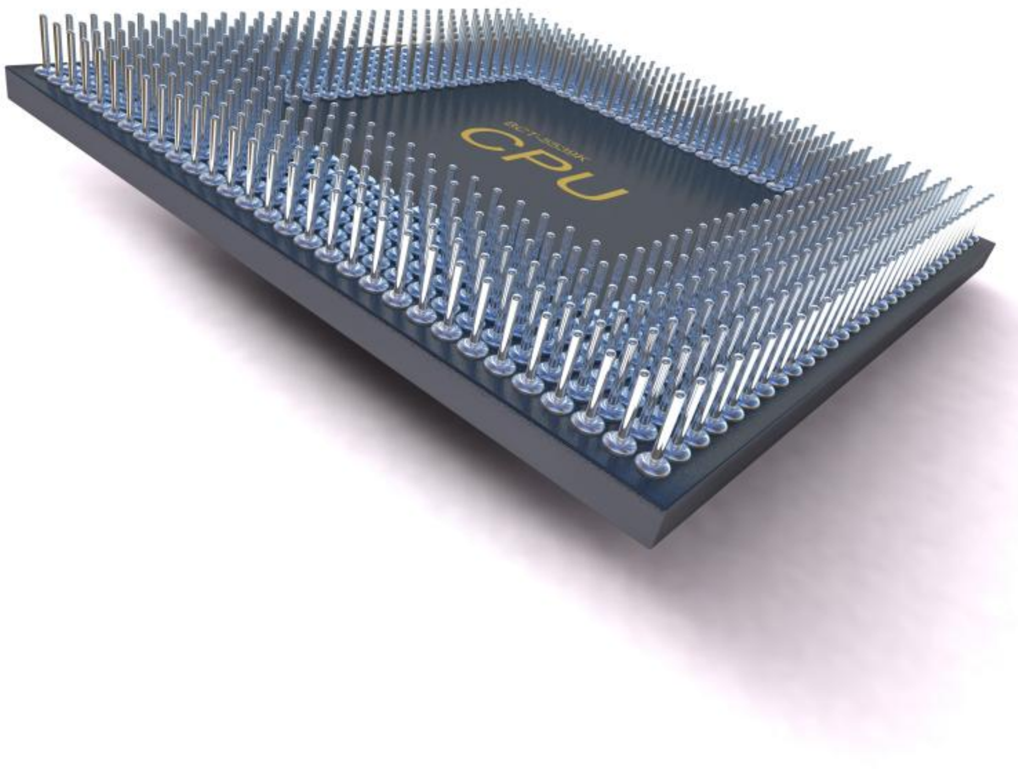
IBM

Microsoft

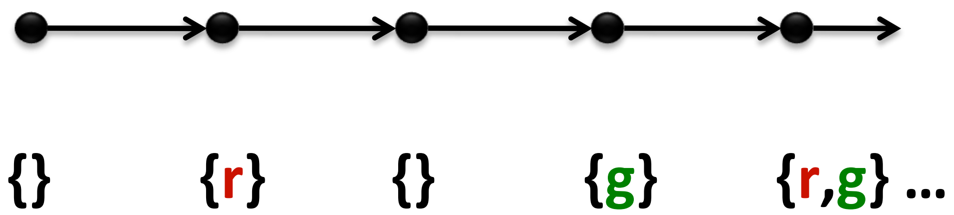
SIEMENS

infineon

OneSpin
SOLUTIONS



An Execution



Representing Sets of Executions

transition system M

(all system executions)

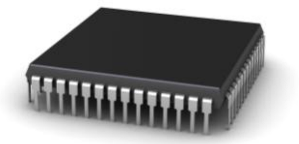
specification

temporal logic formula F

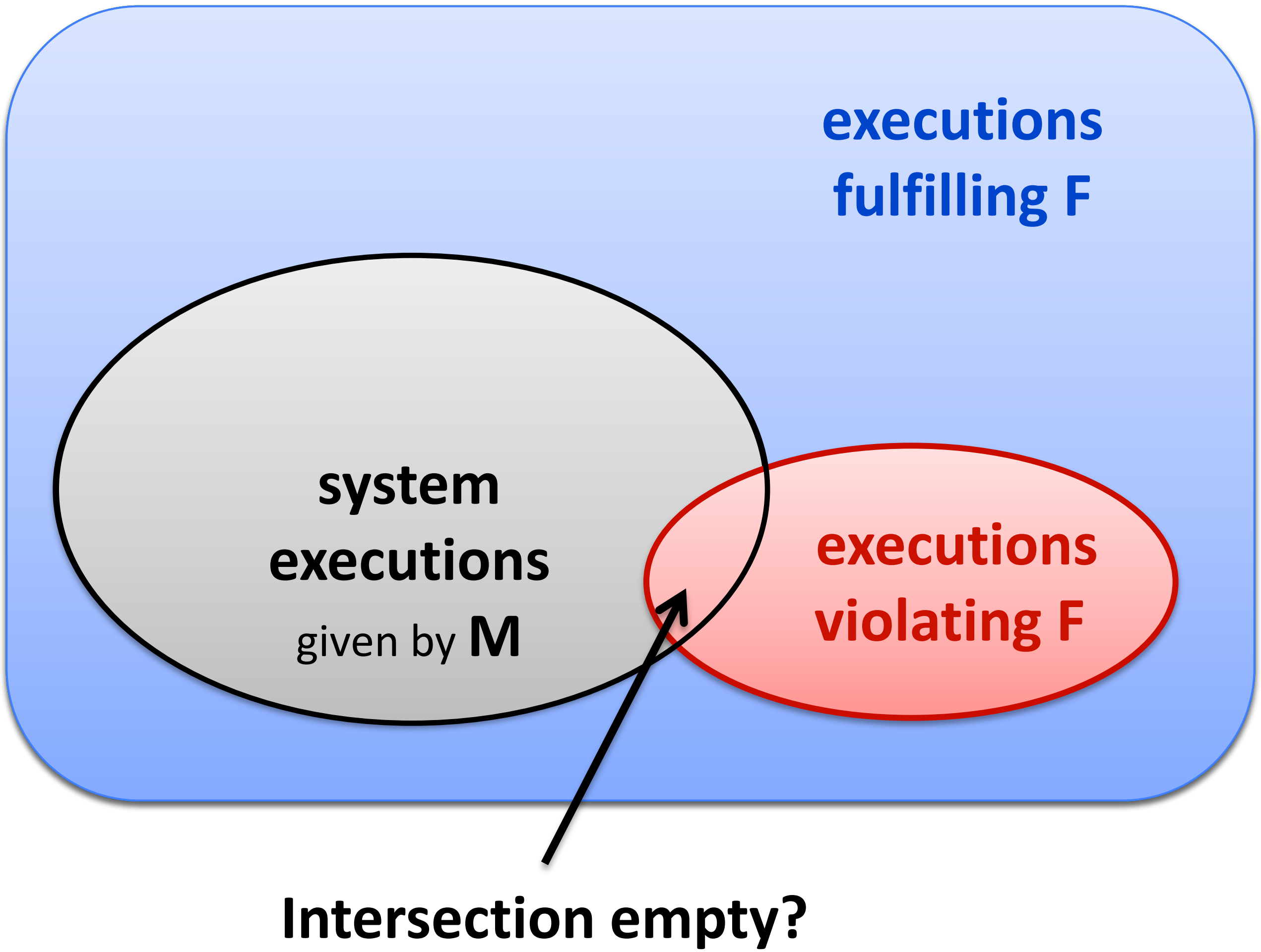
(all allowed executions)

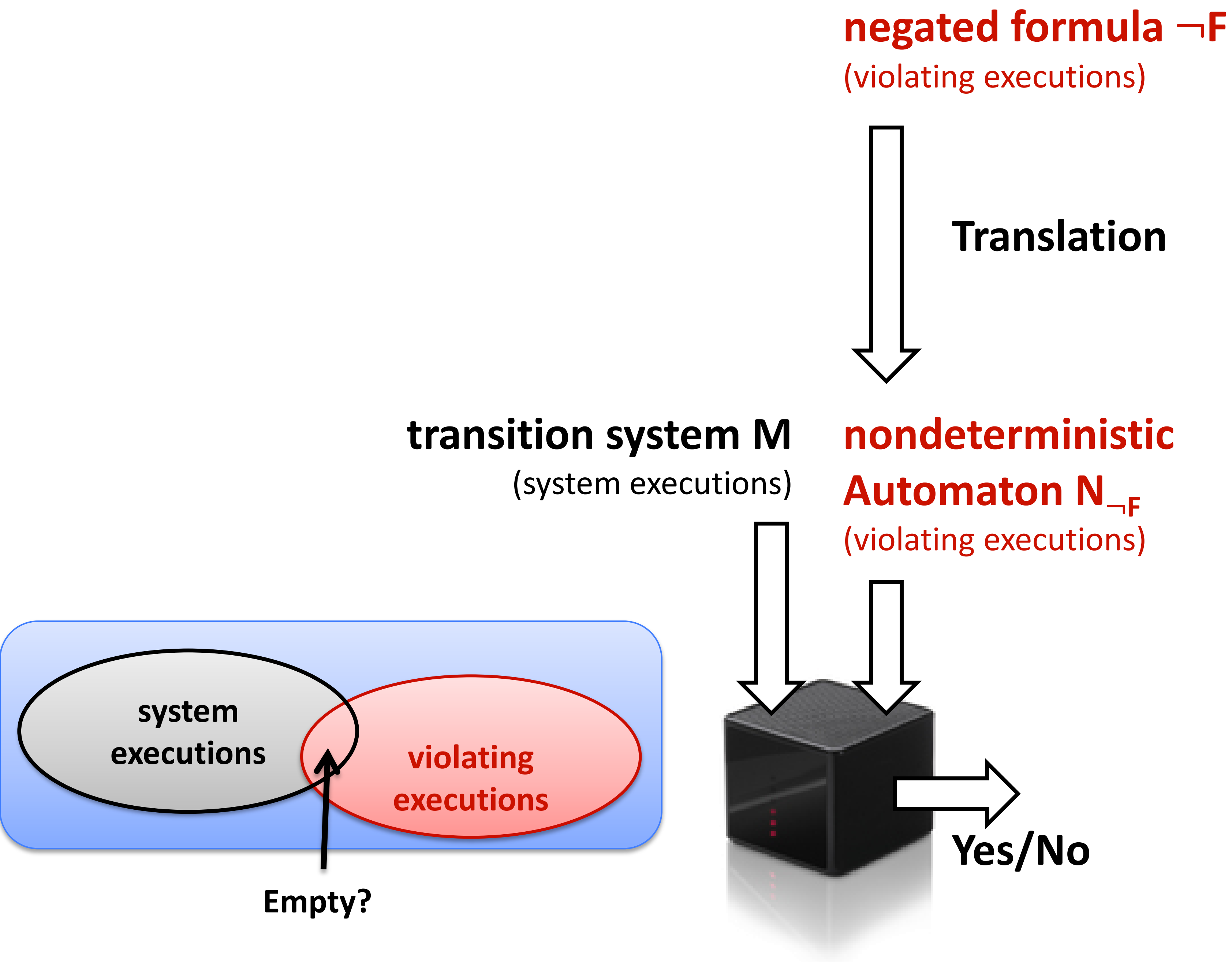
formula F

Generally[g --> $\underbrace{\textbf{Once(r)}}_{\text{r occurs at least once in the past}}$]



{}	{r}	{}	{g}	{r,g} …	**fulfills F**
{r,g}	{}	{g}	{g}	{} …	**fulfills F**
{}	{}	{g}	{}	{} …	**violates F**


executions
fulfilling F
system
executions
given by M
executions
violating F
Intersection empty?

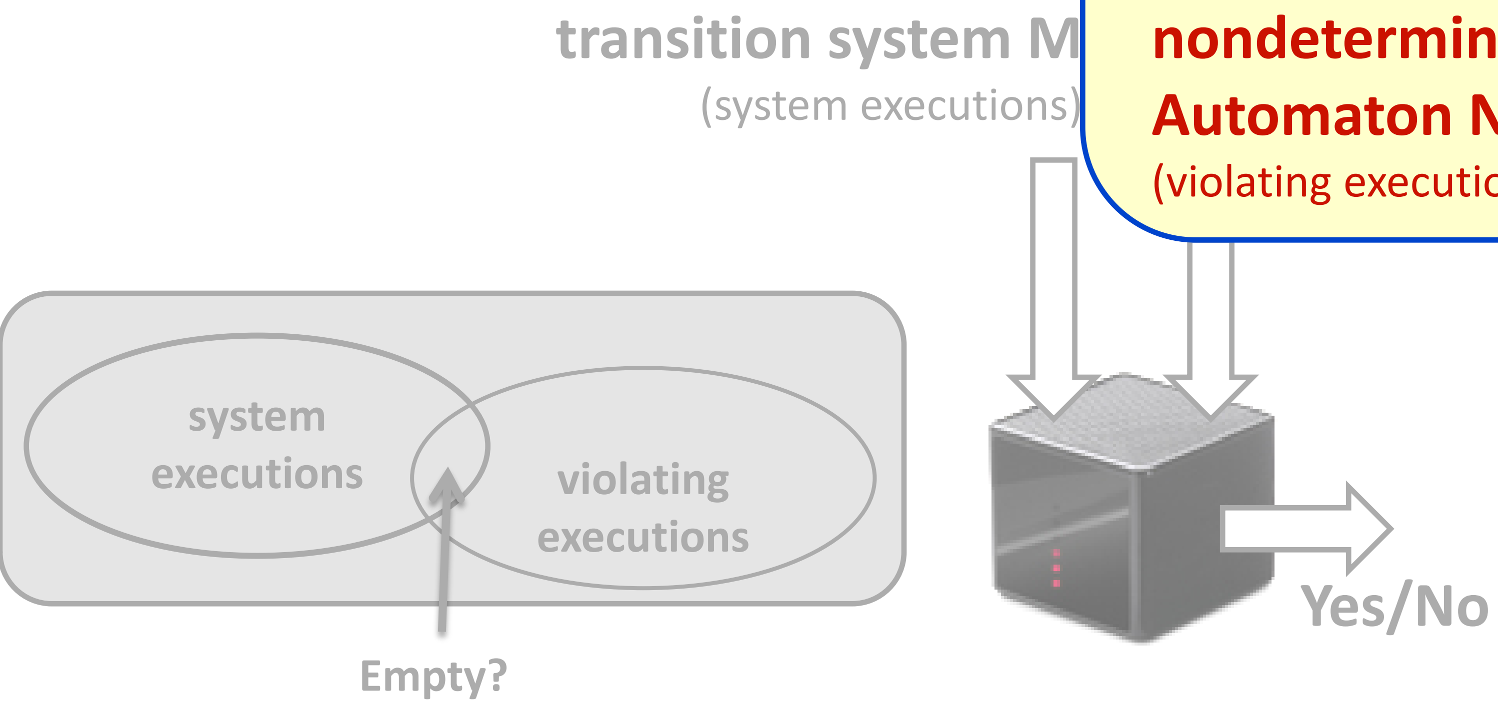

negated formula ¬F
(violating executions)
Translation
transition system M
(system executions)
nondeterministic Automaton $N_{\neg F}$
(violating executions)
system executions
violating executions
Empty?
Yes/No

Scope of Thesis

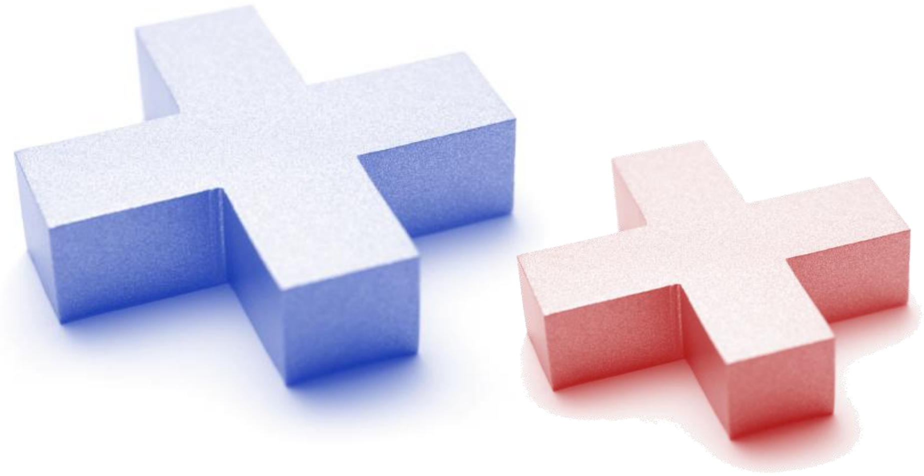
negated formula ¬F
(violating executions)

Translation

nondeterministic Automaton $N_{\neg F}$
(violating executions)



Contributions



PSL + Translation

LTL + Translation

NWTL + Translation

... + Translation

DLTL + Translation

RLTL + Translation

μNWTL + Translation

NWPSL + Translation

μLTL + Translation

CTL* + Translation

First Contribution: Reduction Scheme

formula F

direct, rather simple

alternating automaton A_F

automaton model with rich structure

nondeterministic automaton N_F

Reduction Scheme

- **divide** and conquer
- reduce to **standard constructions** on **nondeterministic** automata

PSL + Translation
instance

LTL + Translation
instance

... + Translation

DLTL + Translation
smaller results

RLTL + Translation
smaller result

μNWTL + Translation
smaller results + more standard

NWPSL + Translation
instance

μLTL + Translation
smaller results

CTL* + Translation

Second Contribution: Complementations

Complementation Constructions

- for **2-way nondeterministic** automata
- based on **standard constructions**

New

+ Reduction Scheme

new translations for logics with **past operators**

Amir Pnueli and others

"… many **statements** that arise naturally in specifications, are **easier to express** using the **past operators**."

[The Glory of the Past '85]

PPSL + Translation

smaller results

PLTL + Translation

more standard

PNWTL + Translation

flaw corrected

... + Translation

PDLTL + Translation

smaller results

PRLTL + Translation

smaller results

μPNWTL + Translation

smaller results + more standard

NWPPSL + Translation

smaller results

μPLTL + Translation

smaller results

PCTL* + Translation

Reduction Scheme

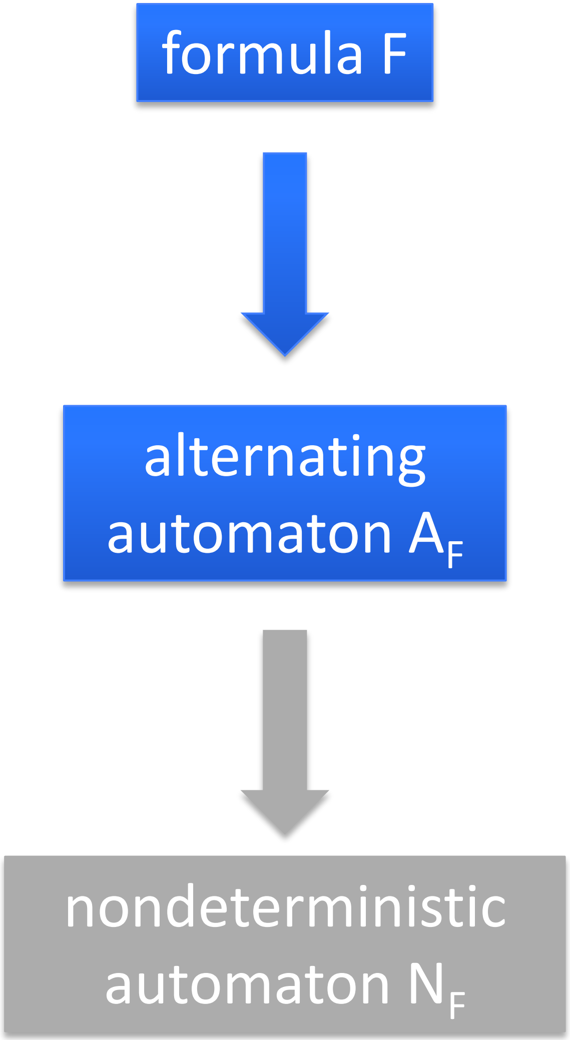
Past Operators

IEEE PSL

Reduction Scheme

Past Operators

IEEE PSL

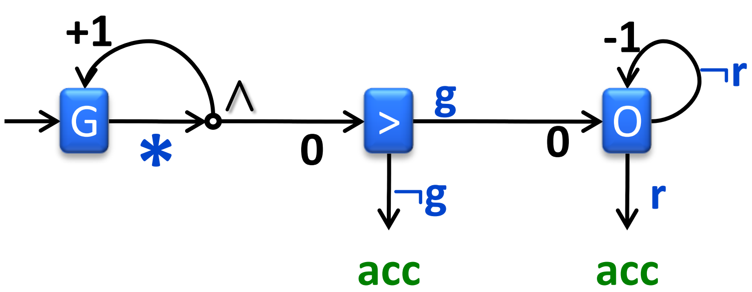

formula F
alternating automaton A_F
nondeterministic automaton N_F

Preliminaries: Alternating Automaton

An 2-way **alternating automaton** is a tuple $(Q, \Sigma, \delta, q_I, \mathcal{F})$

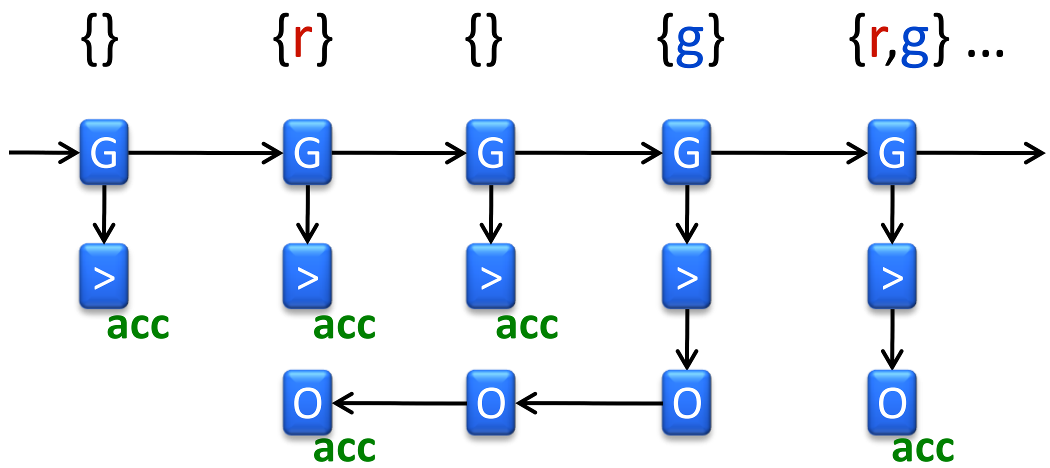
- Q : states, Σ : alphabet, q_I : initial state
- δ: $Q \times \Sigma \to \mathcal{B}^+(Q \times \{-1, 0, 1\})$: transition function
- $\mathcal{F} \subseteq Q^\omega$: acceptance condition

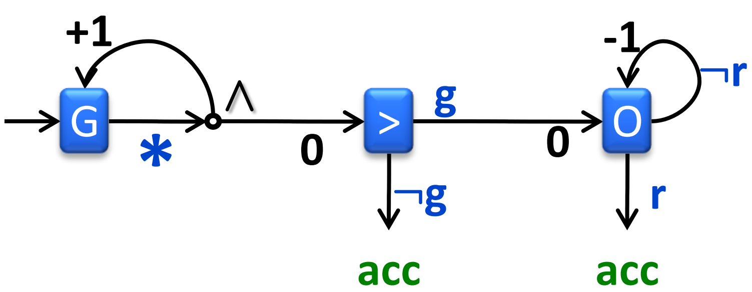
Generally[g --> **Once**(r)]



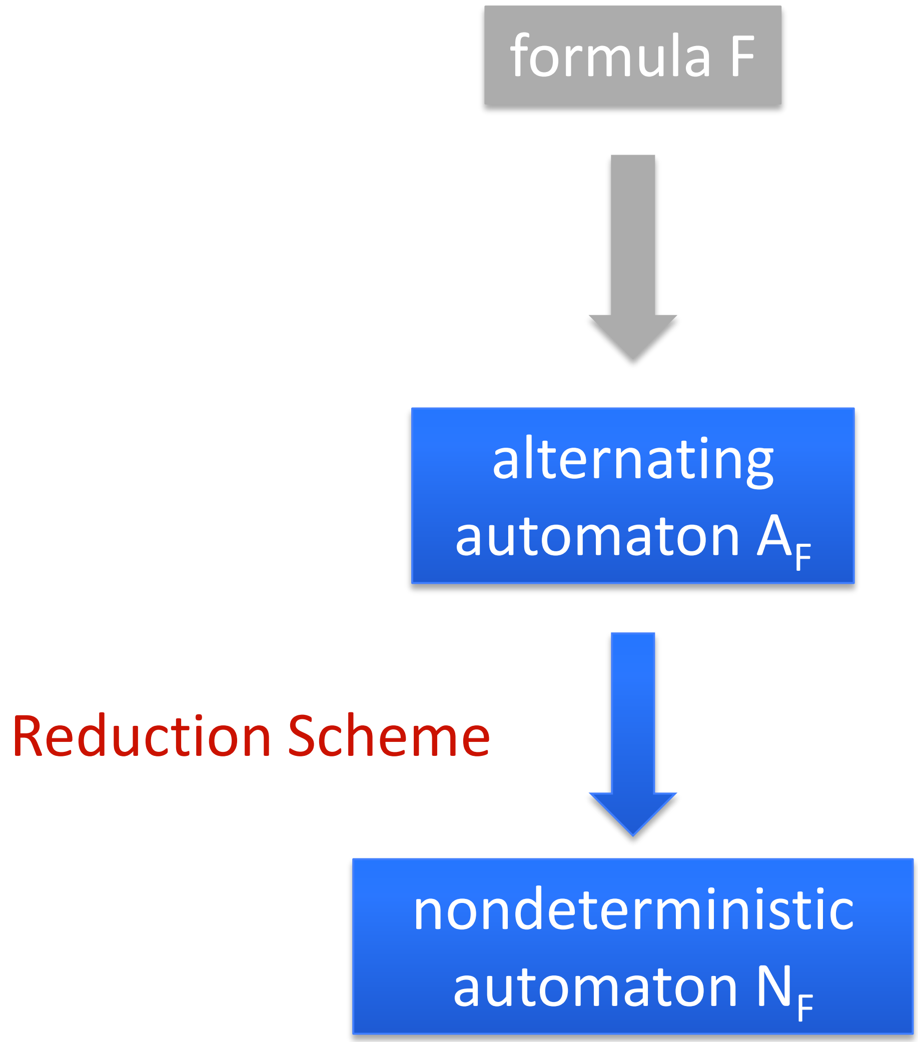
$\mathcal{F} := \{q_0 q_1 \ldots \in \{G, >, O\}^\omega \mid G \text{ occurs } \infty\text{-often}\}$

Preliminaries: Runs by Example





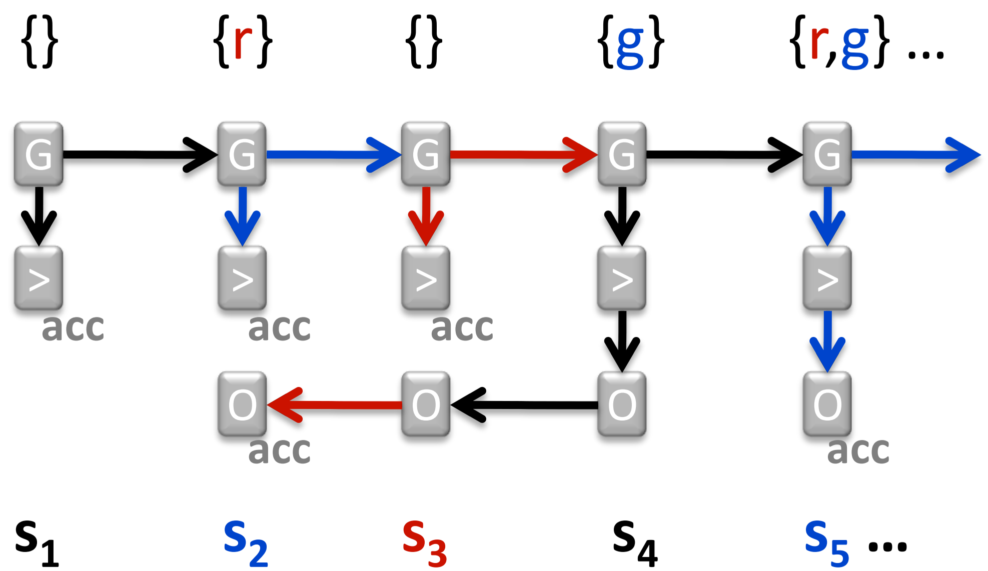
$\mathcal{F} := \{q_0 q_1 \ldots \in \{\mathbf{G}, >, \mathbf{O}\}^\omega \mid \mathbf{G}$ occurs ∞-often$\}$


formula F
alternating automaton A_F
Reduction Scheme
nondeterministic automaton N_F

Prerequisite: Encoding of a Run

Encode run as $s_1 s_2 s_3 \ldots \in (Q \to 2^{Q \times \{-1, 0, 1\}})^\omega$

(sequence of successor functions)



$G \mapsto \{ (G, 1), (>, 0) \}$
$O \mapsto \{ (O, -1) \}$

Reduction Scheme

Given: alternation automaton $\mathcal{A} = (Q, \Sigma, \delta, q_I, \mathcal{F})$

1. **Construct** nondeterministic „**refuter automaton**"
   - $\mathbf{R} := (Q, \Sigma \times (Q \to 2^{Q \times \{-1, 0, 1\}}), \eta, q_I, Q^\omega \setminus \mathcal{F})$
   - $\eta(q, (a, s)) := \begin{cases} s(q) & \text{if } s(q) \text{ satisfies } \delta(q, a), \\ acc & \text{otherwise.} \end{cases}$

   accepts ⇔ tree is **not** accepting run

2. **Complement R**

   accepts ⇔ tree is an accepting run

3. **Project** resulting automaton on Σ

Result: nondeterministic automaton

accepts ⇔ ∃ tree that is an accepting run

POPL ’88 Vardi → ICALP ’98 Vardi → CONCUR ’01 Kupferman, Piterman, Vardi → **Reduction Scheme** New

- Extracts, improves, and generalizes “complementation idea”
- Translations seen as instances:
  - Constructions and proofs become modular
  - Ingredients replaceable by better standard constructions

Reduction Scheme

Past Operators

IEEE PSL

PPSL + Translation

PLTL + Translation

PNWTL + Translation

... + Translation

PDLTL + Translation

PRLTL + Translation

μPNWTL + Translation

NWPPSL + Translation

μPLTL + Translation

PCTL* + Translation

PPSL + Translation

PLTL + Translati

PDLTL + Translation

Complementation of a

nondeterministic + **2-way**

refuter automaton

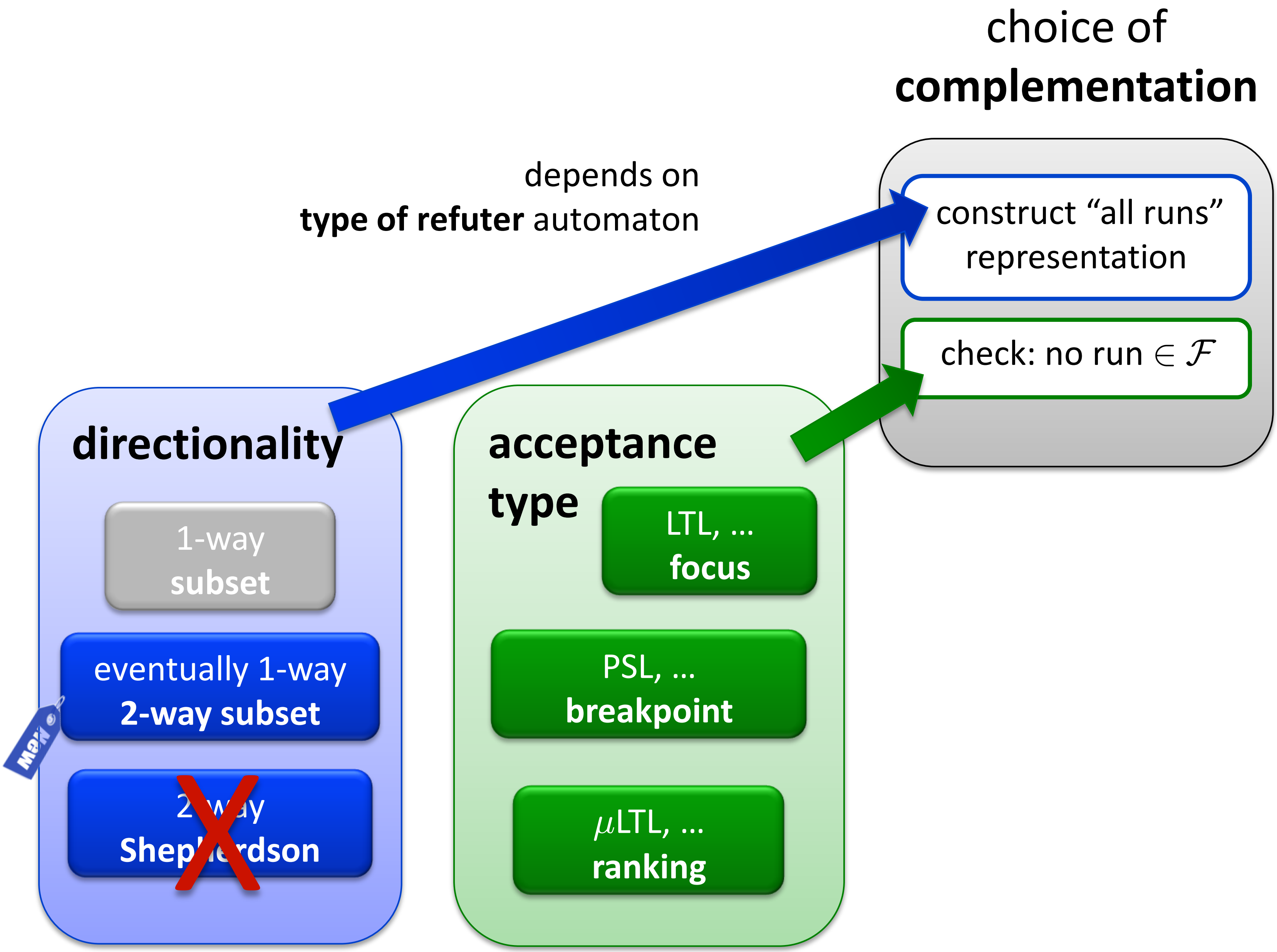
μPNWTL + Translation

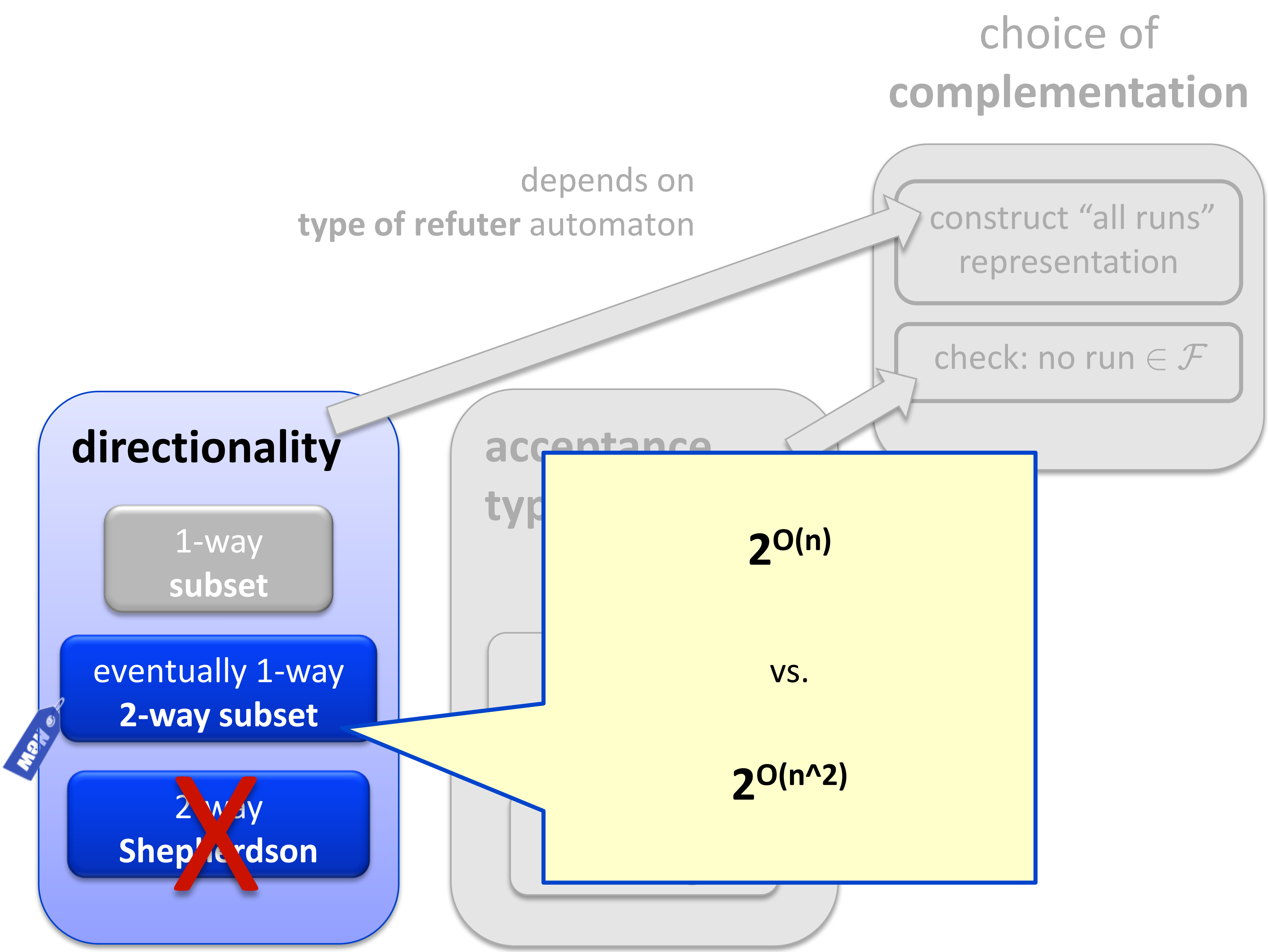
SL + Translation

Reduction scheme identifies

one common ingredient

slation


choice of
complementation
construct "all runs" representation
check: no run $\in \mathcal{F}$
depends on
type of refuter automaton
directionality
1-way
subset
eventually 1-way
2-way subset
2-way
Shepherdson
acceptance type
LTL, ...
focus
PSL, ...
breakpoint
μLTL, ...
ranking



choice of
complementation
depends on
type of refuter automaton
construct “all runs” representation
check: no run $\in \mathcal{F}$
directionality
1-way
subset
eventually 1-way
2-way subset
New
~~2-way~~
~~**Shepherdson**~~
$2^{O(n)}$
vs.
$2^{O(n^2)}$

Reduction Scheme

Past Operators

IEEE PSL

PSL is an IEEE standardized temporal logic

PSL is widely used in hardware industry

PSL consists of LTL + regular expressions

Generally[r Followed_by(Eventually(g))]

r := {start; true*; end} ∩ {¬cancel }*

BUT

PSL has (almost) **no past operators**

"… arbitrary **mixing of past and future** operators results in **nonnegligible** implementation **cost**."

[The ForSpec Temporal Logic '02]

Moshe Vardi and others

however …

We show that

PPSL is exponentially **more succinct** than **PSL**

Cost of translations

	PSL	**PPSL**
Size of **Automaton**	$O(3^{2^n})$	$\cancel{2^{O(2^{2n})}}$ **$O(2^m\, 3^{2^n})$**
Size of **Transition system**	**$O(3^{2^n})$**	**$O(3^{2^n})$**

n := size of formula
m := number of propositions

Conclusion

Reduction Scheme

- Extracts, improves, generalizes "complementation idea"
- Simplifies translations & correctness proofs

Complementation for 2-Way Automata over (nested) words

- Optimized translations for logics with past
- Enables symbolic model checking with PPSL