From Linear-Time Logics to Automata

Christian Dax

ETH Zurich

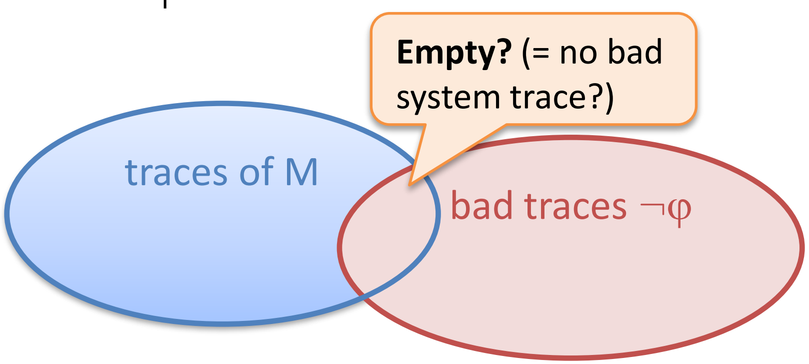
Joint work with

Felix Klaedtke and Martin Lange

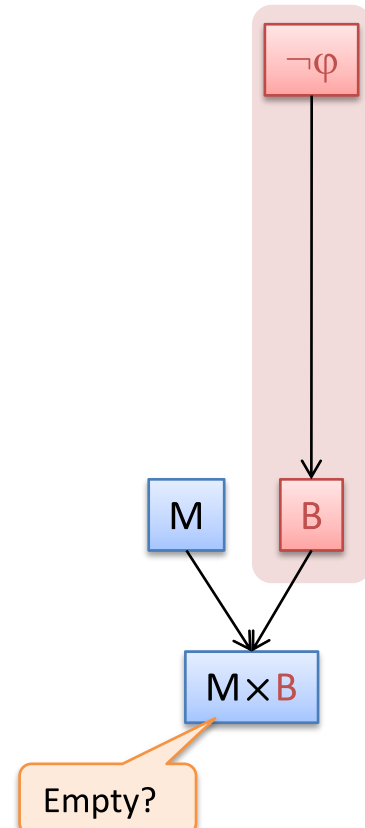
Konstanz, March 2009

Motivation: Finite-State Model Checking

- Model-checking problem:
  - Given: finite-state system M (system traces)
  - Given: specification as formula φ **(good traces)** ⇒ ¬φ (bad traces)
  - Question: $M \models \varphi$?



- Automata-based approach:
  1. Represent sets by **nondeterministic automata**
  2. Represent intersection by product automaton



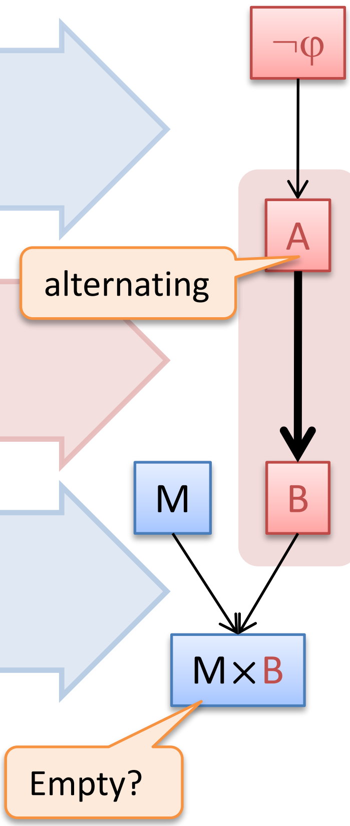
Motivation: Divide and Conquer

- **Alternating automaton** as intermediate step

Step 1
- direct and simple
- linear blow-up

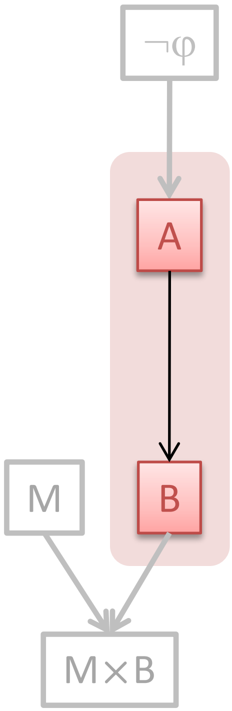
Step 2 -- this talk --
- complex
- exponential blow-up

Further steps
- SPIN model checker
- SMV model checker

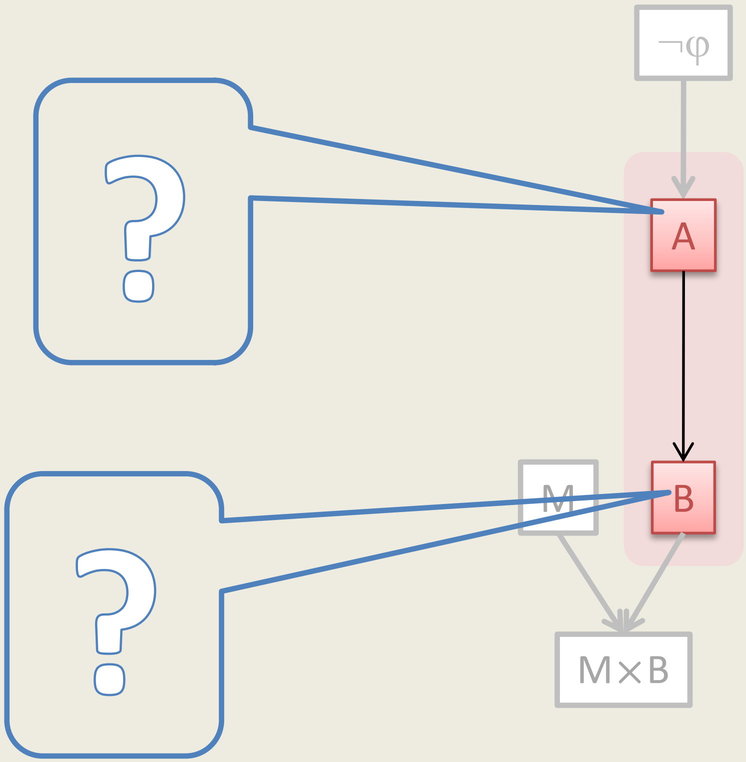


Outline

1. Background
2. **Part I:** Alternation Elimination Scheme
3. **Part II:** Scheme Instance for 2-Way Büchi Automata



Background on Automata

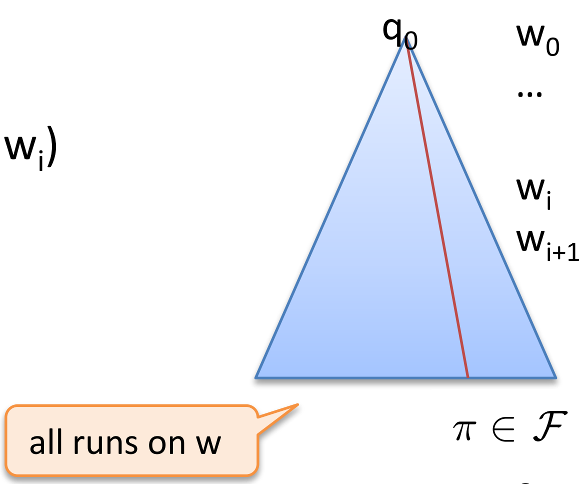


Nondeterministic Automaton (NA)

- An **NA** is a tuple $(Q, \Sigma, \delta, q_0, \mathcal{F})$
  - $\delta: Q \times \Sigma \rightarrow 2^Q$ transition function
  - $\mathcal{F} \subseteq Q^\omega$ acceptance condition
    (state sequences that are considered to be accepting)

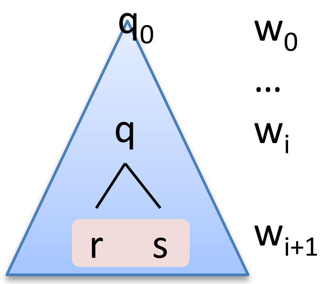
- Remark: **Büchi** and **coBüchi** condition given as $F \subseteq Q$
  - $\mathcal{F} = \{\pi \in Q^\omega \mid$ an F-state occurs ∞-often in $\pi\}$
  - $\mathcal{F} = \{\pi \in Q^\omega \mid$ no F-state occurs ∞-often in $\pi\}$

- For a word $w = w_0 w_1 \ldots$
  - A **run** $q_0 q_1 \ldots$ is a sequence of states with $q_{i+1} \in \delta(q_i, w_i)$
  - w is **accepted** :$\Leftrightarrow$ there is a run on w that is in $\mathcal{F}$



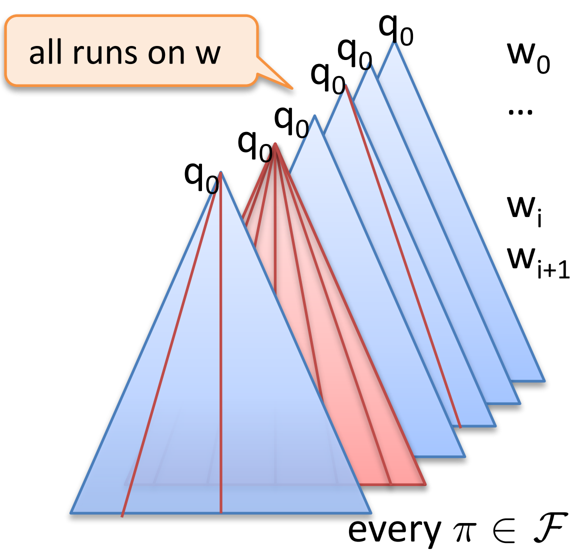
Alternating Automata (AA)

- An **AA** is a tuple $(Q, \Sigma, \delta, q_0, \mathcal{F})$
  - $\delta: Q \times \Sigma \to \mathcal{B}^+(Q)$ transition function
  - $\mathcal{B}^+(Q)$ positive boolean combination of formulas in DNF



$\delta(q, w_i) = (r \land s) \lor (s \land t)$

- For a word $w = w_0 w_1 \ldots$
  - A **run** is a Q-labeled tree, where
    - the root is labeled by q_0, and
    - a q-labeled node in level i has children labeled by states of one of the monomials of $\delta(q, w_i)$
  - w is **accepted**
    $:\Leftrightarrow$ there is a run such that every path is in $\mathcal{F}$



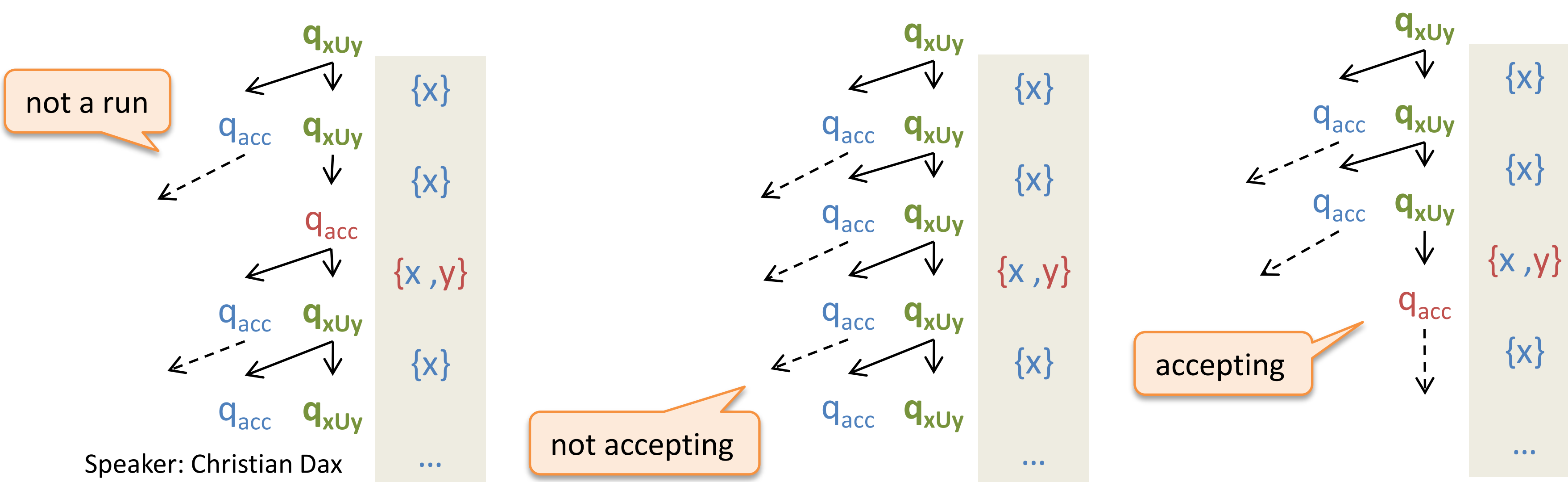
Example: Runs of an Alternating Büchi Automaton

$Q := \{q_{acc}, q_{rej}, q_{xUy}, q_x, q_y\}$, $q_0 := q_{xUy}$, $F := \{q_{acc}\}$

$\delta(q_{xUy}, a) := \delta(q_y, a) \vee (\delta(q_x, a) \wedge q_{xUy})$

$\delta(q_x, a) := \begin{cases} q_{acc} & \text{if } x \in a, \\ q_{rej} & \text{otherwise} \end{cases}$ $\delta(q_y, a) := \begin{cases} q_{acc} & \text{if } y \in a, \\ q_{rej} & \text{otherwise} \end{cases}$

- Some “runs” on the word $\{x\}\{x\}\{x,y\}\{x\}^\omega$

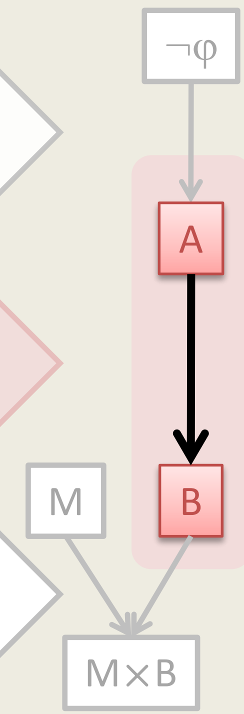


Part I: Alternation Elimination Scheme

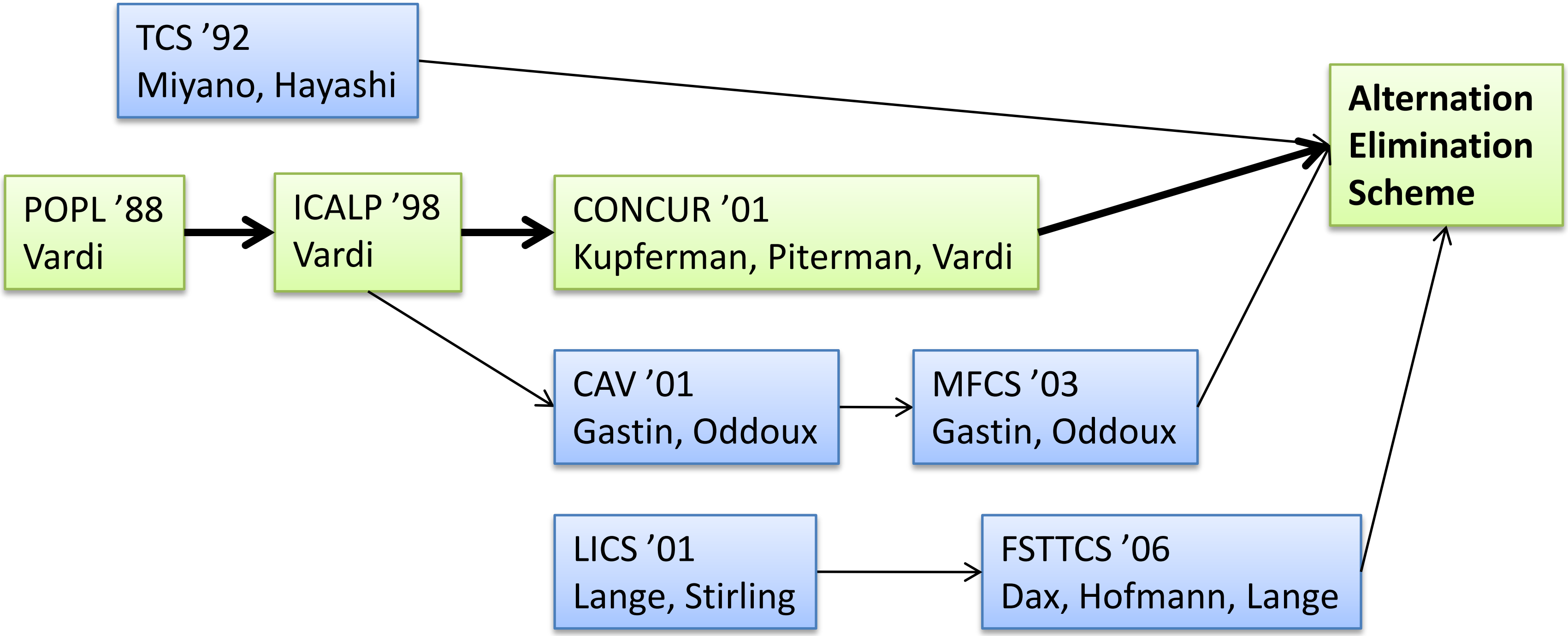
Step 1
- direct and simple
- linear blow-up

Step 2
-- this talk --
- complex
- exponential blow-up

Further steps
- SPIN model checker
- SMV model checker

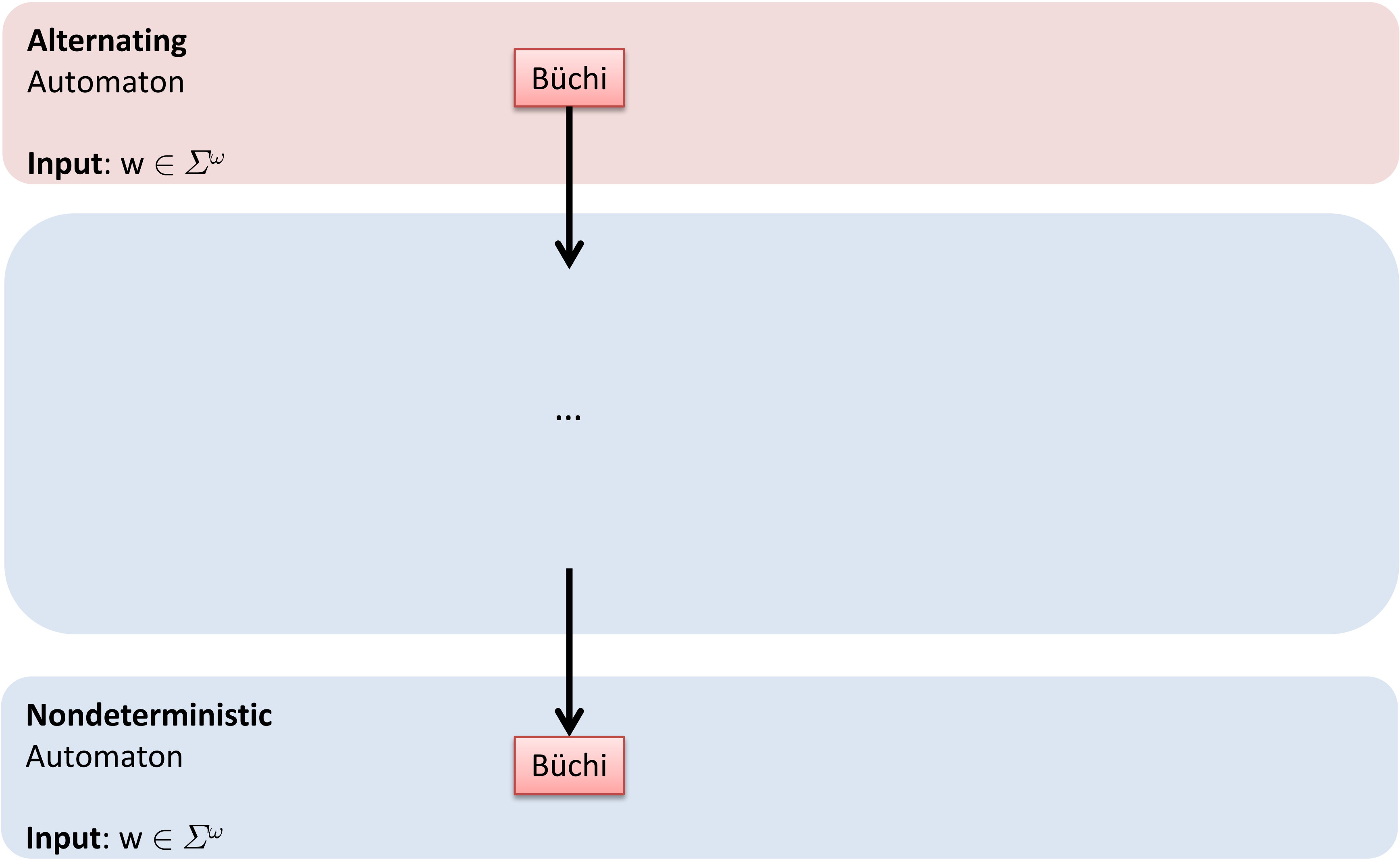


Related Work



- Alternation Elimination Scheme:
  - Improves and generalizes approach used in **green boxes**
  - Unifies + simplifies constructions and proofs of blue boxes that can now be seen as instances

Alternation-Elimination Scheme by Example



Alternation-Elimination Scheme by Example

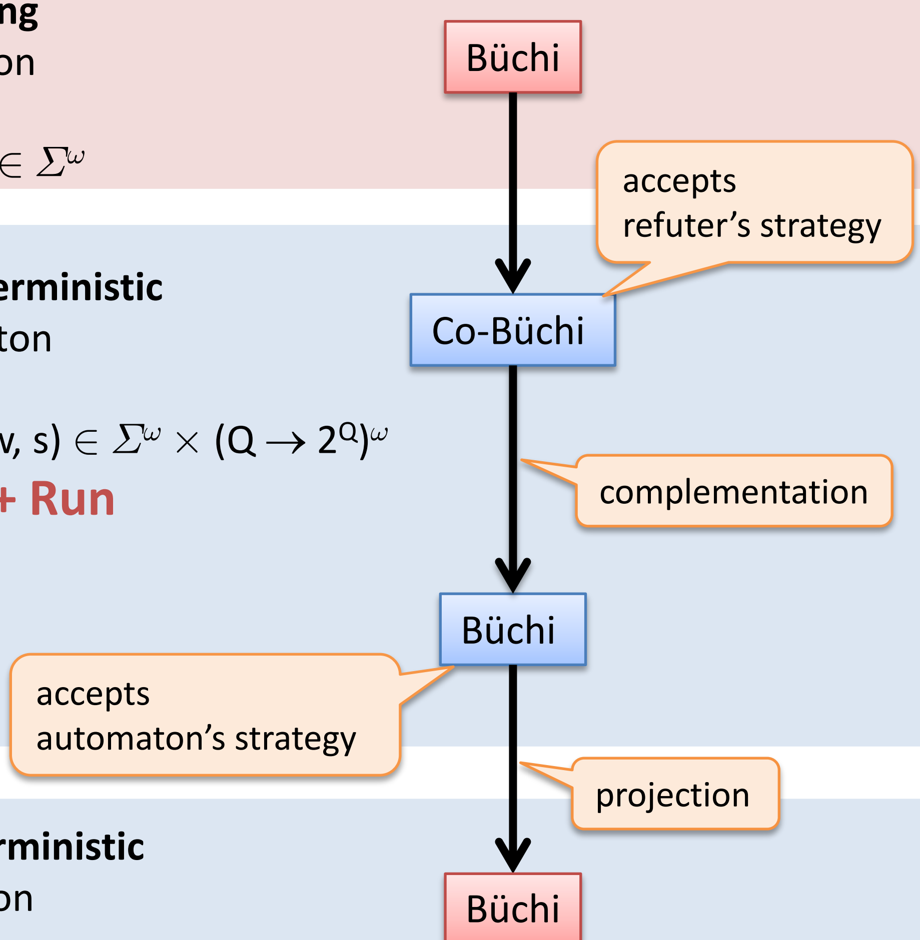
Alternating
Automaton

Input: $w \in \Sigma^\omega$

Nondeterministic
Automaton

Input: $(w, s) \in \Sigma^\omega \times (Q \to 2^Q)^\omega$

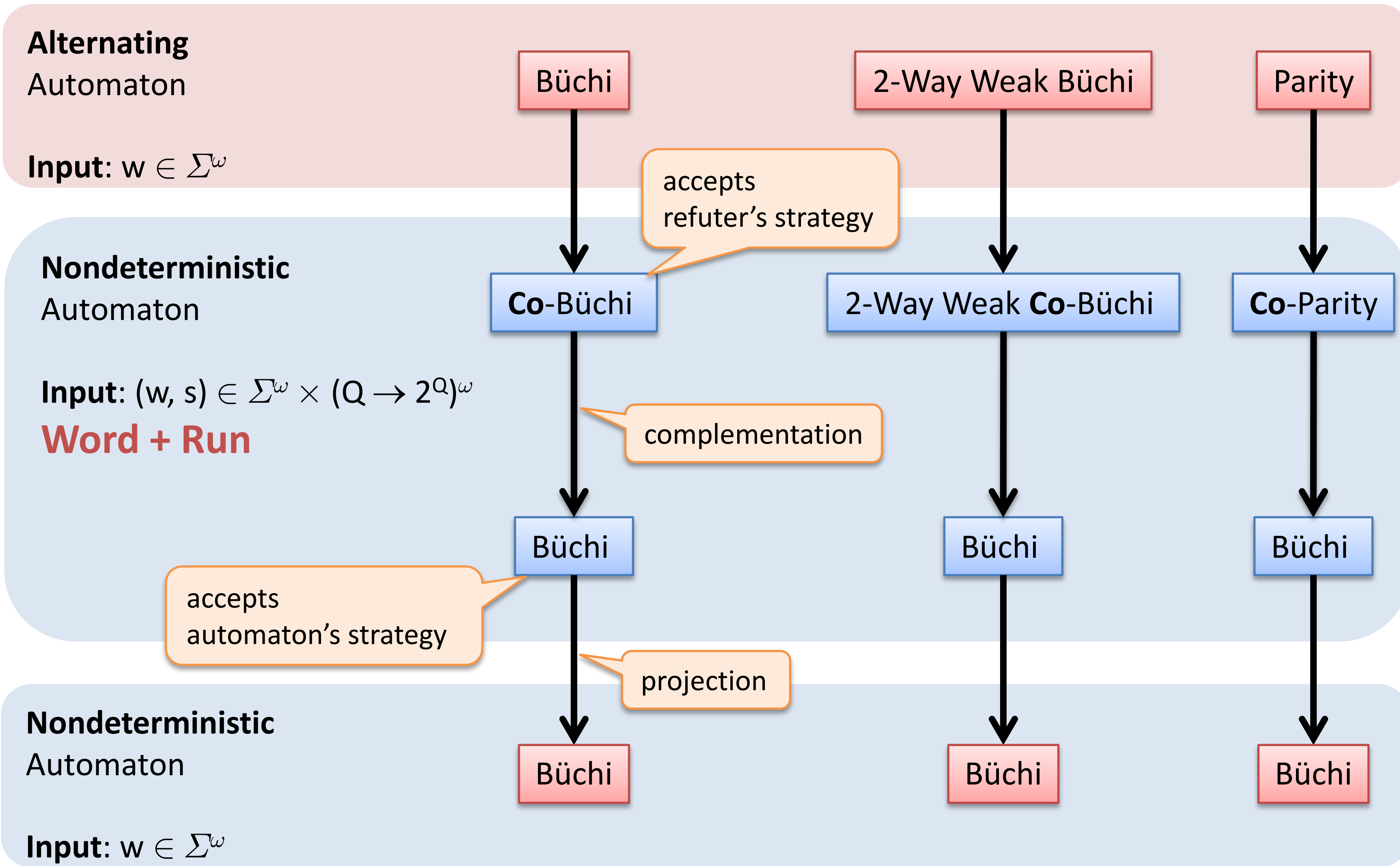
Word + Run



Nondeterministic
Automaton

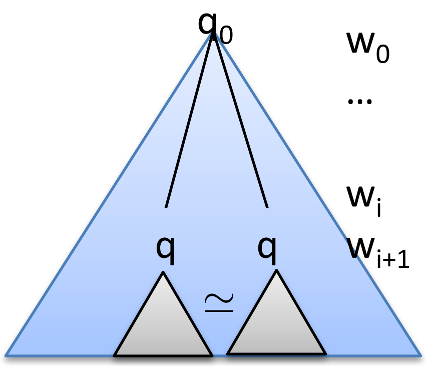
Input: $w \in \Sigma^\omega$

Alternation-Elimination Scheme by Example



Alternation Elimination (1/2) : Runs as Words

- We consider only automata with memoryless runs
  - Examples: Büchi, co-Büchi, Parity, Rabin automata
  - Equally-labeled nodes have equally-labeled subtrees
  - "for equally-labeled nodes, automaton chooses same transition"

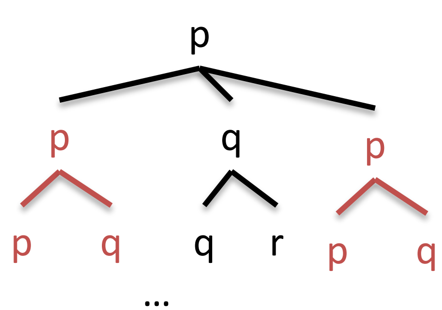


- Encode run as sequence $s_0 s_1 s_2 \ldots \in (Q \to 2^Q)^\omega$ of successor functions
  - $s_i(q)$: 'labels of children of q-labeled node in level i'
  - Example:

$s_0(p) = \{p, q\}$

$s_1(p) = \{p, q\}$, $s_1(q) = \{q, r\}$

$s_2(p) = \ldots$, $s_2(q) = \ldots$, $s_2(r) = \ldots$

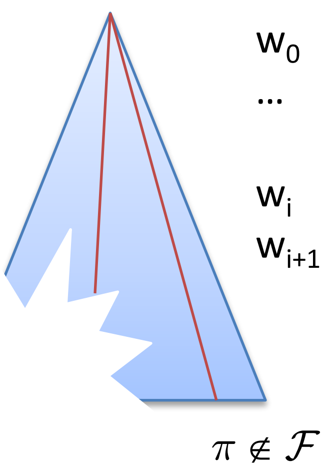


Alternation Elimination (2/2) : Complementation

- Let $\mathcal{A}$ = (Q, Σ, δ, q_0, $\mathcal{F}$) be an AA and Γ := Q $\rightarrow$ 2^Q
- $\mathcal{A}$ accepts the word w
  - $\Leftrightarrow$ there is a run on w s.t. every path is in $\mathcal{F}$
  - $\Leftrightarrow$ $\exists$s: s $\in$ runs(w) $\wedge$ $\forall\pi \in$ s: $\pi \in \mathcal{F}$
  - $\Leftrightarrow$ $\exists$s: $\neg$ (s $\notin$ runs(w) $\vee$ $\exists\pi \in$ s: $\pi \notin \mathcal{F}$) ★
  - $\Leftrightarrow$ $\exists$s: $\neg$ $\mathcal{B}$(w, s)

'refuter's strategy'

- It is easy to build an NA $\mathcal{B}$ over $\Sigma\times\Gamma$ for ★
  - $\mathcal{B}$:= (Q, $\Sigma\times\Gamma$, η, q_0, $Q^\omega \setminus \mathcal{F}$)
  - $$\eta(q, (a, s)) := \begin{cases} s(q) & s(q) \text{ is a monomial in } \delta(q, a) \\ \{q_{acc}\} & \text{otherwise} \end{cases}$$



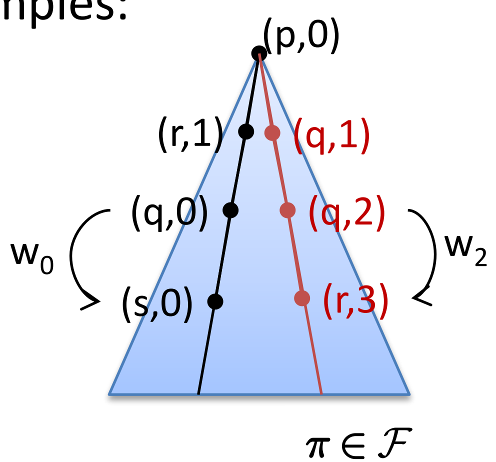
- Finally: complement the NA $\mathcal{B}$ and project it on Σ.

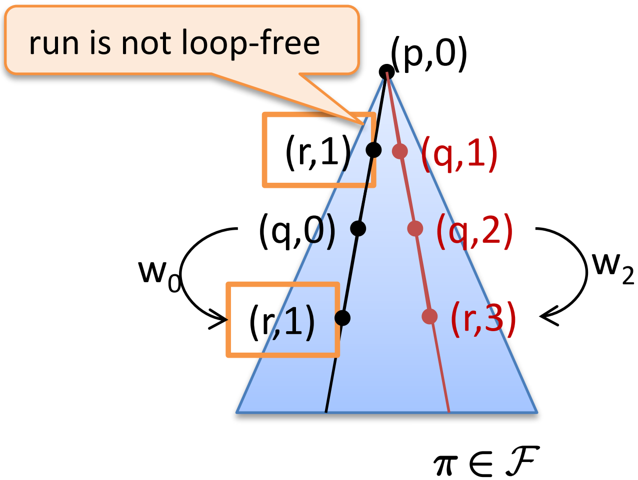
Scheme for 2-Way Automata

- In our paper: scheme also works for **2-way automata**
  - 2-way automata can move the read-only head in both directions.
  - Configuration consists of a state and the position of the read-only head

- Loop-freeness
  - A run is **loop-free** :⇔ for every path, no configuration occurs twice
  - An AA is **loop-free** :⇔ every run is loop-free

- Examples:





Some Instances: Translations to NBAs

- Resulting sizes of 1-way NBAs:

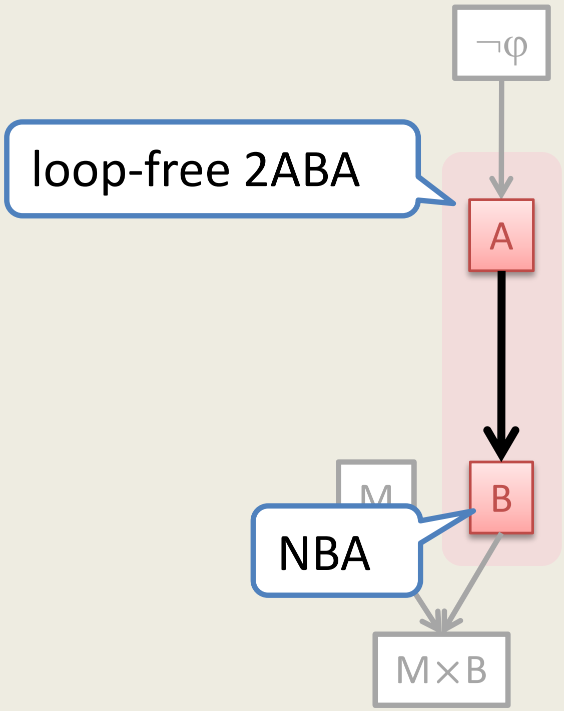
alternating automata

old/new	1-Weak ABA LTL (+ Past)	ABA PSL (+ Past)	AParityA μLTL (+ Past)	ARabinA
1-way	$O(n2^n)$/○	$O(2^{2n})$/○	$2^{O(nk \log n)}$/●	--/**$O(2^{nk \log nk})$**
2-way	--/**$O(n2^{3n})$**	$2^{O(n*n)}$/●	$2^{O(nk*nk)}$/●	
2-way + loop-free	$O(n2^{2n})$/○	--/**$O(2^{4n})$**	--/unpublished	--/unpublished

Part II of this talk

● **smaller constant** (hidden in O notation)
○ **same size** but construction **more modular**

Part II: From 2-Way ABA to NBA



Motivation: From PastPSL to NBA

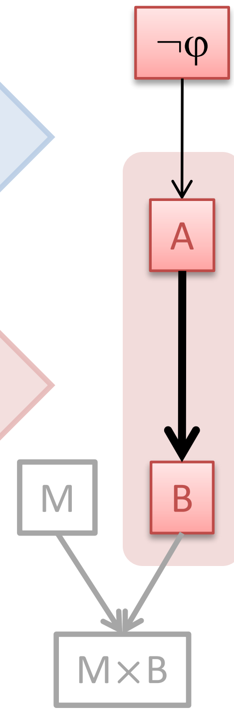
- PastPSL $\simeq$ extension of linear-time logics PSL and SVA with past operators

Step 1

- Translation:
  PastPSL $\longrightarrow$ 2-way ABA

Step 2

- Translation:
  2-way ABA $\longrightarrow$ 1-way NBA

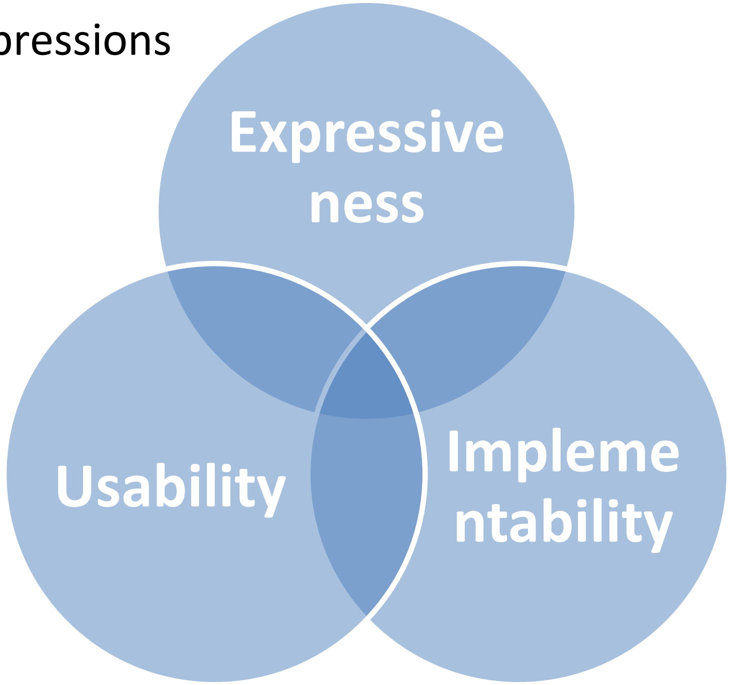


Motivation: Overview on PSL and SVA

- IEEE standardized temporal logics
  - linear core of PSL = LTL + semi-extended regular expressions
  - linear core of SVA = semi-extended regular expressions

- Widely used in hardware industry (Intel, IBM, Infineon, …)

- Well balances between competing needs of specification languages:
  - **Expressiveness**: omega-regular languages
  - **Usability**: formulas are easy to read and write
  - **Implementability**: model-checking problem is solvable in practice



Motivation: Why Past Operators?

- PSL and SVA have no past operators
  - Justification: **"… arbitrary mixing of past and future operators results in nonnegligible implementation cost."** [TACAS'02]

- **However:**
  - Past Operators for LTL are natural to express properties like

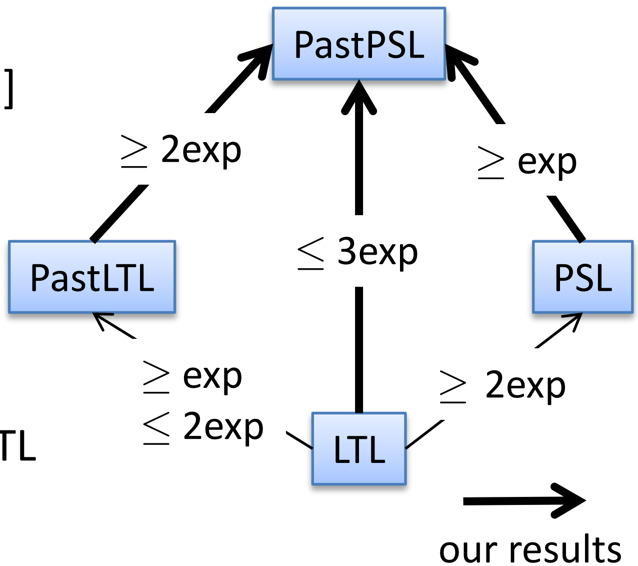
    G(**grant** → O **request**)

  - Another example:
    - Every **grant** is preceded by a **request**.
    - **request** = **start** followed by an **end** with no cancel in between

    G(**grant** → O {{**start**; true*; **end**} ∩ {¬cancel}*})

Motivation: Why Past Operators?

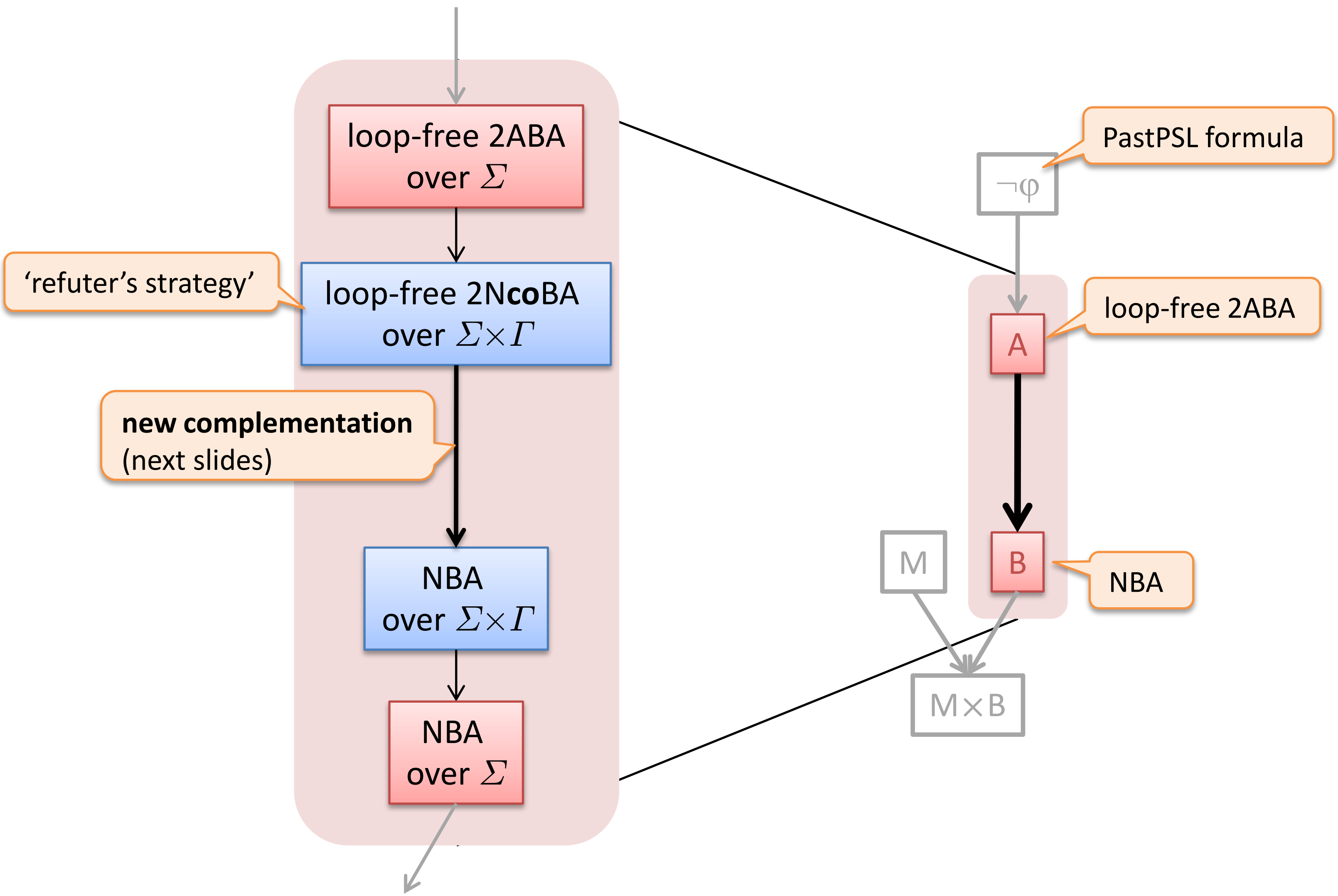
- PSL and SVA have no past operators
  - Justification: **"… arbitrary mixing of past and future operators results in nonnegligible implementation cost."** [TACAS'02]



- **However:**
  - PastPSL is
    - exponentially more succinct than PSL and SVA,
    - double-exponentially more succinct than PastLTL
  - Implementation cost is negligible in theory and does not exist in practice for symbolic model checking.

$|NBA_{PastPSL}| = O(2^{3 * 2^{2n}})$ **vs.** $|NBA_{PSL}| = O(2^{2 * 2^{2n}})$

Outline: From PastPSL to NBA

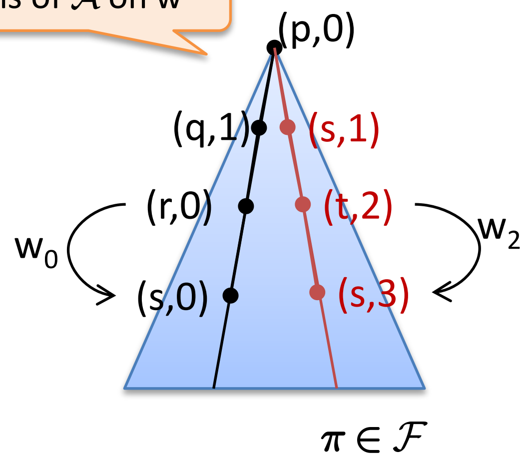


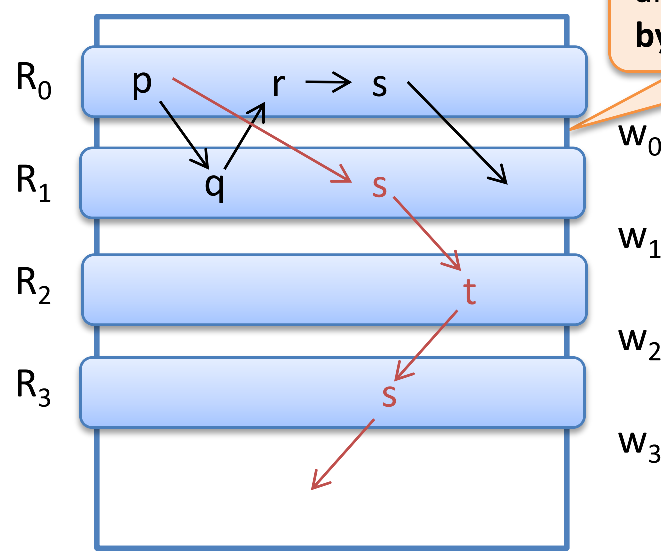
Complementation Construction

- Given loop-free 2NcoBA $\mathcal{A}$ = (Q, Σ, δ, q_0, F)
  1. $\mathcal{A}$ **accepts** w $\Leftrightarrow$ $\exists$ **run** on w: **no F-state occurs ∞-often**
  2. $\mathcal{A}$ rejects w $\Leftrightarrow$ $\forall$ run on w visits an F-state ∞-often
- Construct NBA that checks 2.
  - Guess sequence $R_0R_1 \ldots \in (2^Q)^\omega$
  - Check that sequence is consistent with δ

"2-way powerset construction"

all runs of $\mathcal{A}$ on w

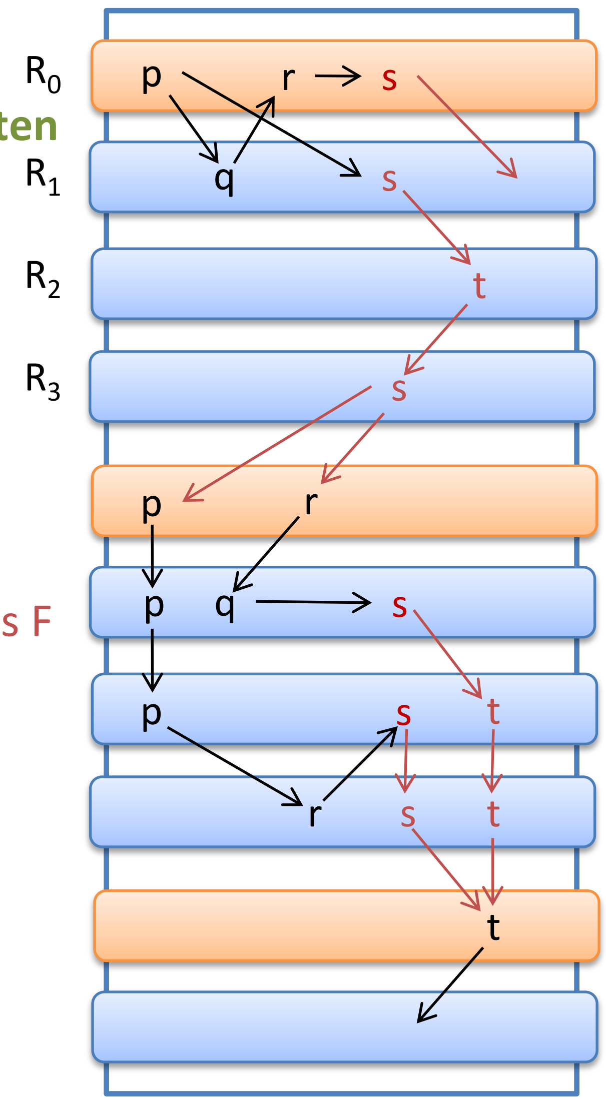




all runs on w **ordered by head position**

Complementation Construction

- Given loop-free 2NcoBA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$
  1. $\mathcal{A}$ accepts w $\Leftrightarrow \exists$ run on w: no F-state occurs ∞-often
  2. $\mathcal{A}$ rejects w $\Leftrightarrow \forall$ run on w visits an F-state ∞-often
- Construct NBA that checks 2.
  - Guess sequence $R_0R_1 \ldots \in (2^Q)^\omega$
  - Check that sequence is consistent with δ
  - Guess breakpoints:
    - each run starting at the previous breakpoint visits F before reaching the next breakpoint
    - Example: F = {s}
  - Check that breakpoints occur ∞-often.



Conclusion

- Alternation-elimination scheme
  - Requires complementation construction for NA with co-acceptance condition
  - Novel constructions generalizes known constructions and proofs
  - Also works for tree automata, visibly pushdown automata, …
- PastPSL
  - Novel efficient symbolic translation to NBAs
  - Succinctness results for PastPSL with respect to PastLTL and PSL
  - Past operators and 2-way automata are not difficult
- Ongoing and future work: Implementation
  - Unexpectedly good results for symbolic model checking
  - Work in progress for SPIN model checking