Mechanizing the Powerset Construction for Restricted Classes of ω-Automata

Christian Dax[1], Jochen Eisinger[2], Felix Klaedtke[1]

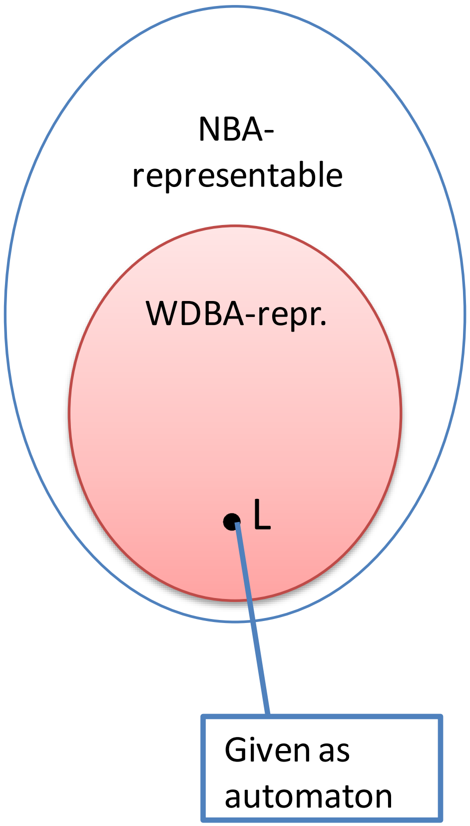
[1]ETH Zurich

[2]Albert-Ludwigs-University of Freiburg

ATVA 2007

Motivation

- Reasoning with restricted classes of nondeterministic Büchi automata (NBA) is often more efficient:
  - Weak deterministic Büchi automaton (WDBA): complement in O(n), minimize in O(n log n)
- How to get a WDBA for language L if
  - L is given as automaton and
  - L can be represented by WDBA?
- Practical relevance:
  - Deciding FO($\mathbb{R}$,$\mathbb{Z}$,<,+)
  - Model checking of WDBA-representable specifications



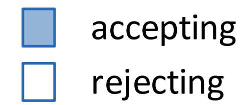
Outline

- Determinization construction for obtaining a WDBA

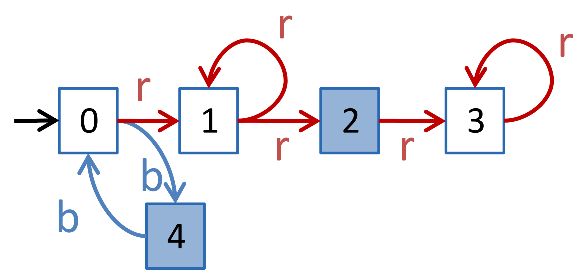
- Applications:
  - Deciding FO($\mathbb{R}$,$\mathbb{Z}$,<,+)
  - Model checking WDBA-representable specifications

Determinization construction for obtaining a WDBA

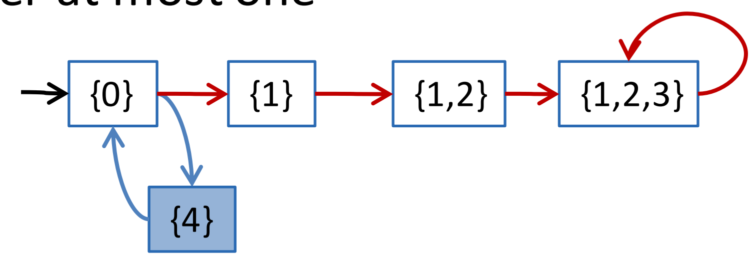
Background: NBAs, DBAs, WDBAs



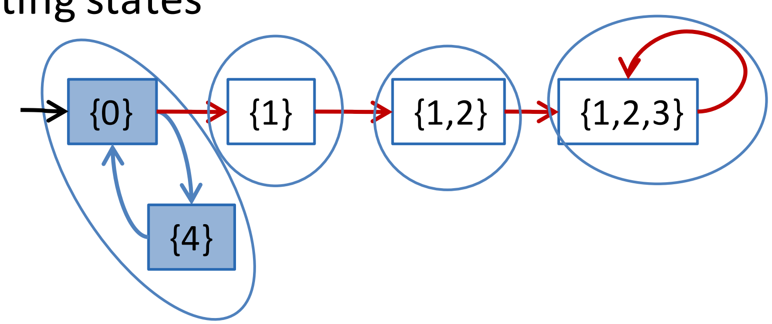
- Σ={r,b}. NBA accepts only "bbb…"



- Deterministic (DBA): for each letter at most one successor state

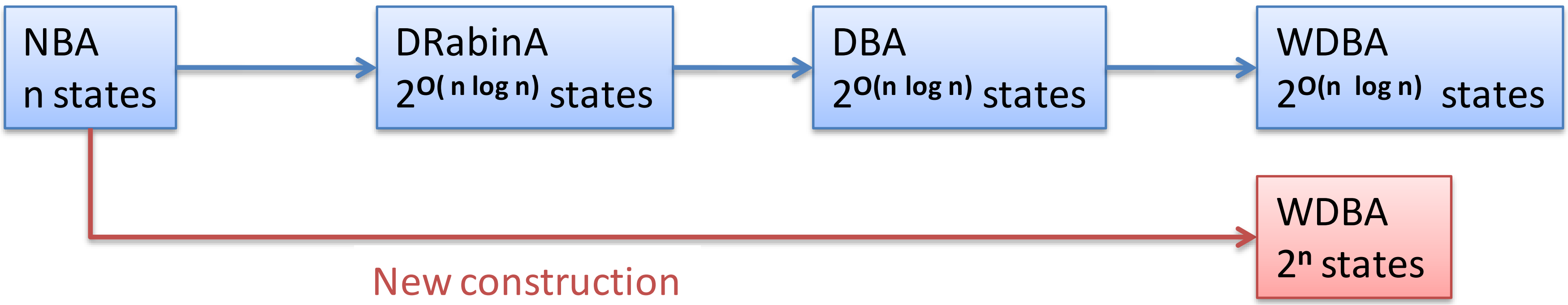


- Weak + Deterministic (WDBA)
  - Weak: each maximal strongly connected component (SCC) has only accepting or only rejecting states



Advantage of new construction

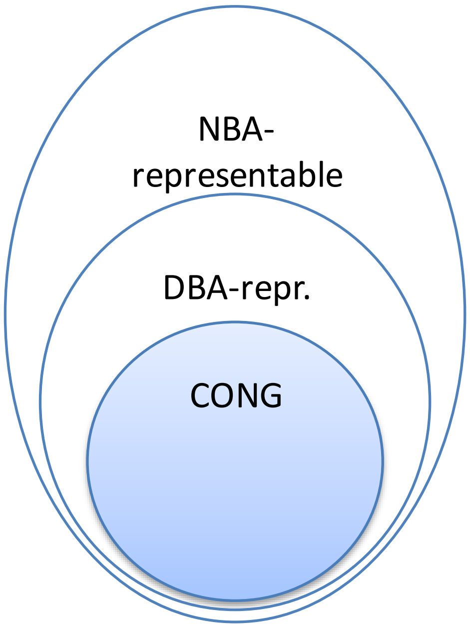
- Goal: For given automaton, we want WDBA representation
- Best alternative construction we know of:



- Advantages:
  - usually has fewer states + worst case slightly better
  - easier to implement: powerset vs. Safra trees
  - works also for Parity/Muller/Rabin/… as input automata

The language class CONG

- Σ^*/Σ^ω set of all finite/infinite words over Σ
- For $L \subseteq \Sigma^\omega$: syntactic right-congruence $\approx_L \subseteq \Sigma^* \times \Sigma^*$,
  - Similar to Myhill-Nerode congruence on finite words
  - $u \approx_L v$ iff $\forall w \in \Sigma^\omega$: $uw \in L \Leftrightarrow vw \in L$

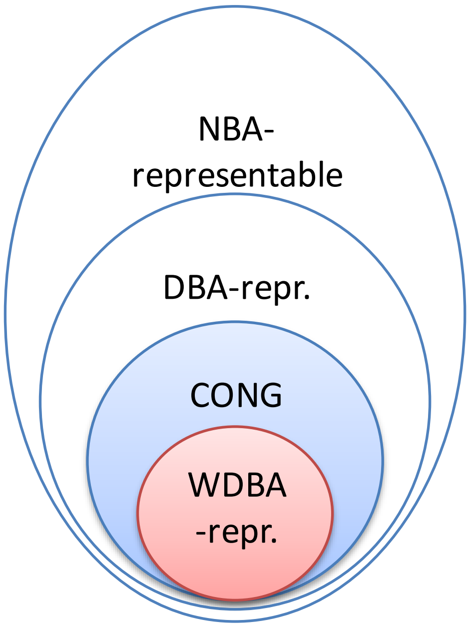


- $\mathcal{C}_L = (Q, \Sigma, \delta, q_I)$ congruence transition system for $\approx_L$:
  - Classes of $\approx_L$ as states: $Q = \{[v] : v \in \Sigma^*\}$
  - Transitions: $\delta([v], a) := [va]$
  - Initial state: $q_I = [\varepsilon]$

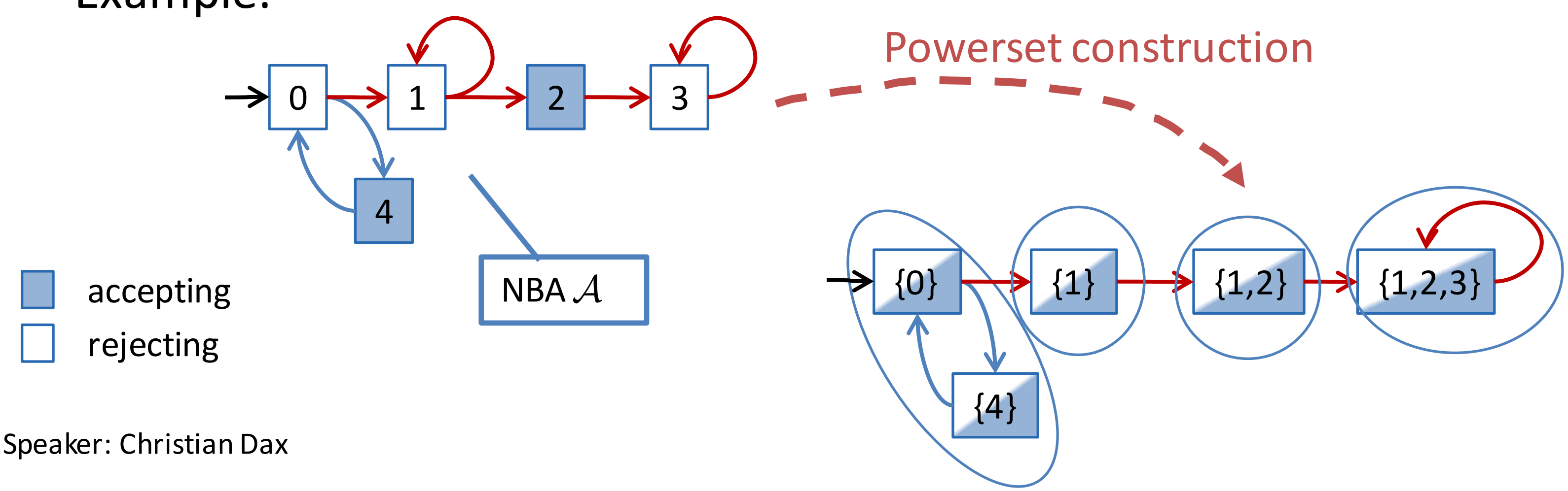
- Automaton = Transition system + accepting states
- CONG := $\{ L \subseteq \Sigma^\omega \mid L(\mathcal{C}_L, F) = L$, for some $F\}$

NBA $\rightarrow$ WDBA determinization

- [our paper] For L $\in$ CONG: Any automaton that represents L can be determinized with the powerset construction
- [Maler, Staiger] Languages that can be represented by WDBAs are in CONG
- [our paper] Algorithms to determine accepting states of WDBAs and DBAs

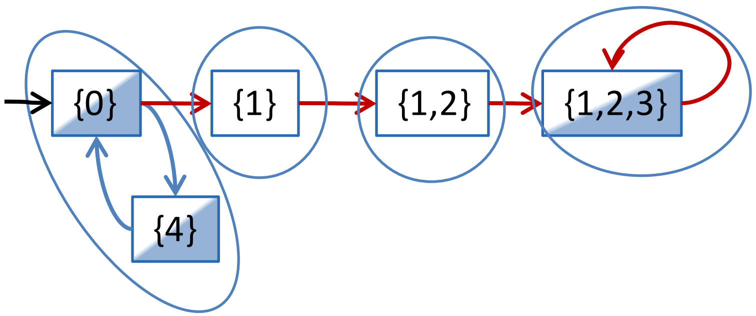


- Example:



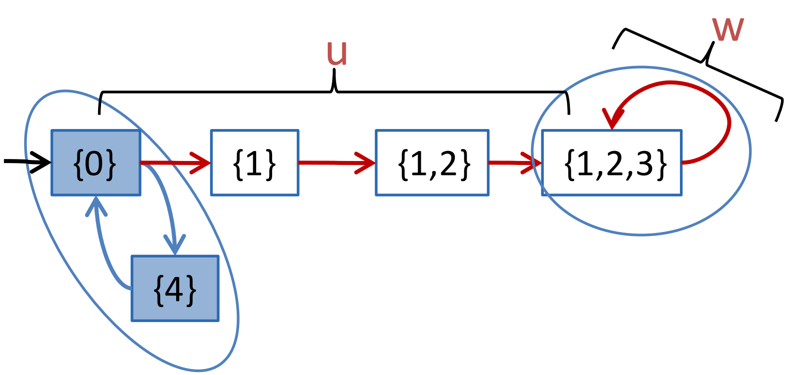
Determine accepting SCCs

- SCC without a loop:
  - Make rejecting
  - Does not change language



- SCC with a loop $w \in \Sigma^*$:
  - $u \in \Sigma^*$ word that induces run into SCC
  - SCC accepting $\Leftrightarrow$ NBA accepts uw^ω

  - NBA $\mathcal{A}$ accepts only "bbb…"
  - (rrr) $(r)^\omega$ rejected by $\mathcal{A}$
  - ε $(bb)^\omega$ is accepted by $\mathcal{A}$



Application 1:
Deciding FO($\mathbb{R}$,$\mathbb{Z}$,<,+)

Decision procedure for FO($\mathbb{R}$,$\mathbb{Z}$,<,+)

- [Boigelot, Wolper] For a given FO($\mathbb{R}$,$\mathbb{Z}$,<,+) formula,
  - WDBA is recursively constructed over the formula structure
  - Formula satisfiable $\Leftrightarrow$ WDBA not empty

- Recursive construction:
  - $\vee$ and $\wedge$ $\Rightarrow$ build product WDBA
  - $\neg$ $\Rightarrow$ complement WDBA
  - $\exists$ quantifier $\Rightarrow$
    - Guess satisfying assignment of quantified variable: WNBA
    - "Breakpoint determinization": WNBA $\rightarrow$ WDBA

- Advantages of new "powerset" over "breakpoint" determinization:
  - 15%—20% memory savings
  - 15%—20% speed up

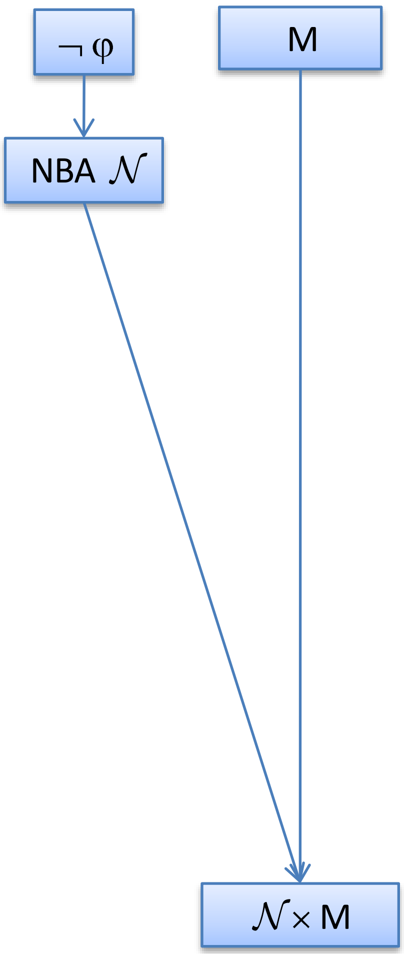
Application 2:
Model checking WDBA-representable specifications

LTL model checking

- Model checker automatically determines whether a system fulfills a specification

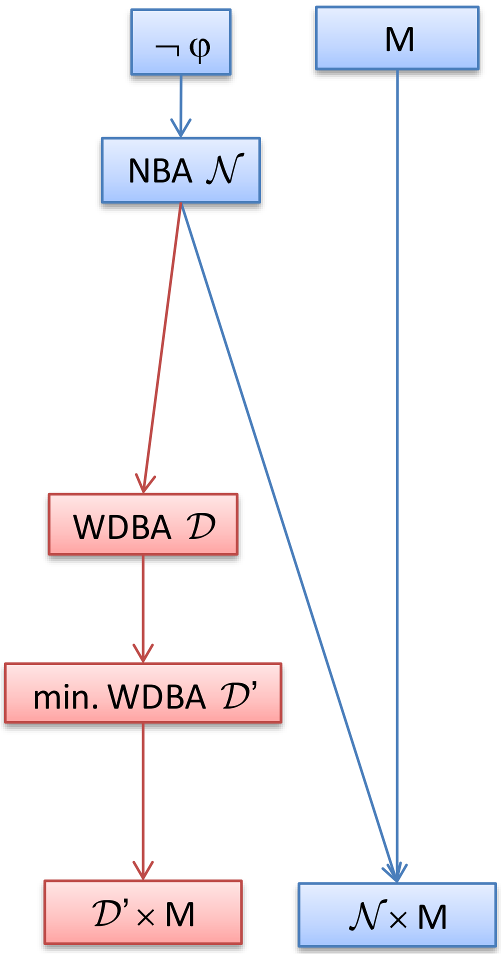
- SPIN model checker:
  - System M: labeled transition system
  - Specification φ: LTL formula
  - Negated specification translated to NBA $\mathcal{N}$
  - M fulfills $\varphi \Leftrightarrow$ no execution in $\mathcal{N} \times$ M (emptiness check)

- Remarks:
  - M is huge.
  - Emptiness check of $\mathcal{N} \times$ M is bottleneck.



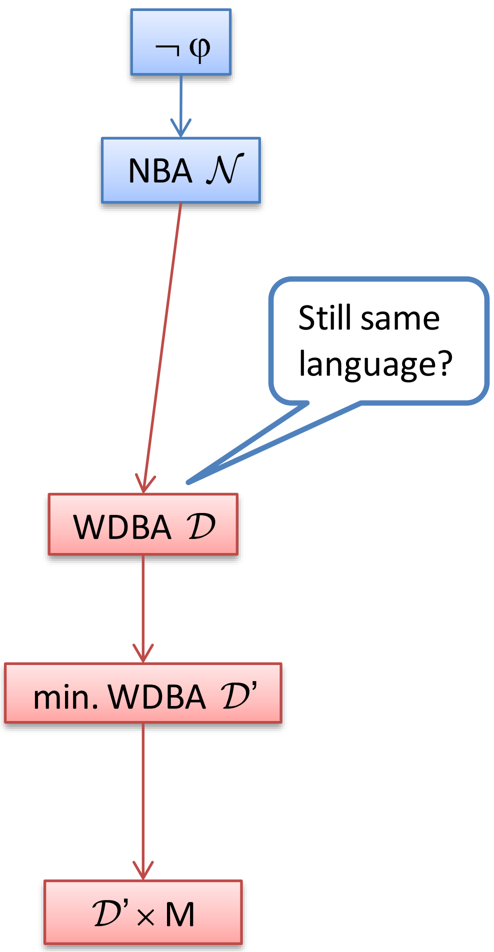
Our approach

- Optimizations to reduce size of $\mathcal{N} \times$ M:
  - Reduce size of $\mathcal{N}$
  - [Sebastiani, Tonetta] Heuristics to make $\mathcal{N}$ "more deterministic"
- Our approach for WDBA-representable specifications:
  - Powerset determinization: $\mathcal{N} \rightarrow \mathcal{D}$
  - Checking that L($\mathcal{N}$) = L($\mathcal{D}$)
  - Minimization: $\mathcal{D} \rightarrow \mathcal{D}'$
- Related: Translation into finite word automata for safety specifications [Vardi, Kupferman, Lampert]



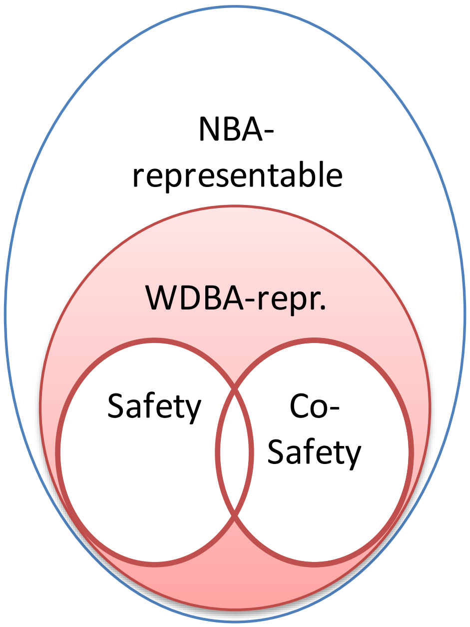
Is the specification WDBA-representable?

- Check that $L(\mathcal{D}) = L(\mathcal{N})$
  - $\Leftrightarrow L(\mathcal{D}) \subseteq L(\mathcal{N})$ and $L(\mathcal{N}) \subseteq L(\mathcal{D})$
  - $\Leftrightarrow L(\mathcal{D} \times \neg\mathcal{N}) = \emptyset$ and $L(\mathcal{N} \times \neg\mathcal{D}) = \emptyset$
  - Complement WDBA $\mathcal{D}$: $\neg\mathcal{D}$
  - Translate formula to NBA $\neg\mathcal{N}$
- Additional check: safety or co-safety?
  - [Landweber] Safety property $\Leftrightarrow$ min. WDBA has at most one rejecting state, which is a rejecting sink
  - Co-safety: dual check
- Related: Syntactic check whether formula describes safety property [Sistla]



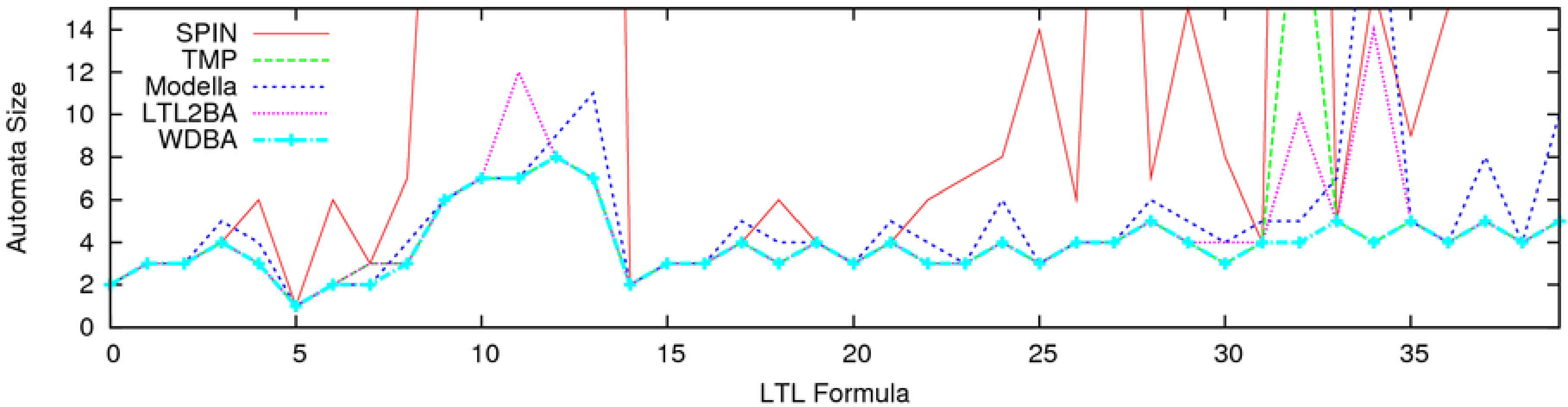
WDBA-representable formulas

- Survey on formulas from the literature:
  - 59 out of 94 are WDBA representable
  - Many formulas specify safety properties. Similar observations by [Cerna, Pelanek]
  - Boolean combinations of safety properties can be represented by WDBAs. [Chang, Manna, Pnueli]



Experimental evaluation I

- First experiment: comparing automata sizes
- Test cases:
  - SPIN, TMP, Modella, LTL2BA: LTL $\longrightarrow$ NBA translators
  - WDBA: LTL2BA + determinization + minimization.
  - 40 formulas from http://patterns.projects.cis.ksu.edu/: templates for commonly used specifications.
- Results:
  - For all formulas: WDBAs sizes not larger than NBAs
  - For formula 34: three times smaller than smallest constructed NBA



Experimental evaluation II

- Second experiment: comparing time/memory usage for emptiness check
- Test cases:
  - Bobdb models an audio/video power controller
  - Elevator2 models an elevator controller
  - Giop models the General Inter-ORB Protocol in CORBA
  - Signarch models an architecture for administrating digital signatures
- Result:
  - Approach with WDBAs is faster and uses less memory

	bobdb		elevator2		giop		signarch	
SPIN	14m04	2.9 GB	--	>3 GB	--	>3 GB	17m57	2.0 GB
TMP	13m53	2.9 GB	7m19	2.2 GB	0m04	0.4 GB	14m25	2.0 GB
LTL2BA	14m04	2.9 GB	7m16	2.1 GB	0m15	0.5 GB	14m23	2.0 GB
MODELLA	14m04	2.9 GB	6m41	2.2 GB	--	>3 GB	14m09	2.0 GB
WDBA	**8m05**	**2.1 GB**	**6m31**	**2.0 GB**	**0m06**	**0.4 GB**	**5m17**	**0.8 GB**

Conclusion

- Contributions
  - Novel determinization construction for automata, whose languages are WDBA-representable.
  - Integration and evaluation of new construction for deciding FO($\mathbb{R}$,$\mathbb{Z}$,+,<): faster + memory savings.
  - Utilization and evaluation of new construction for model checking WDBA-representable specifications: faster + memory savings.
- Future work
  - Tailoring the emptiness check for weak automata.
  - Utilize construction for SAT-based model checking.