NFA vs DFA Representations for the Regular Constraint in Lazy Clause Generation: A Comparative Evaluation

Ross Mackay | r.s.mackay@student.tudelft.nl
Supervisors: Emir Demirović, Imko Marijnissen
Delft University of Technology, The Netherlands

1. Background

Constraint Programming

- Models combinatorial problems as variables with finite domains, and constraints on which value combinations are allowed.
  - A solution assigns every variable a value, satisfying all constraints.
- Solvers prune the search space through **propagation**, repeatedly removing values that can't appear in any solution.
  - Solvers branch when propagation stalls, and backtrack on a dead end.

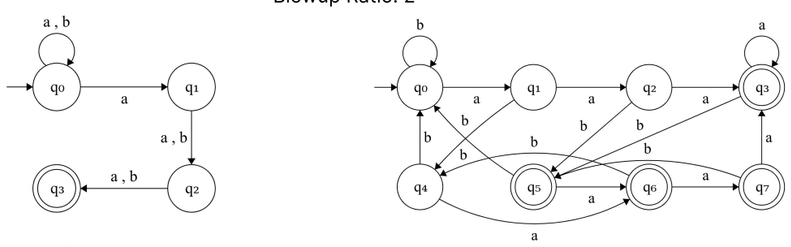
Lazy Clause Generation

- Augments CP propagation with **clause learning**.
  - Every removed value must come with an explanation, based on the assignments that forced the removal.
- Upon conflict, explanations combine to form a learned clause (**nogood**).
  - Recording the cause lets the solver avoid similar conflicts and backtrack non-chronologically, with smaller explanations giving stronger clauses.

The Regular Constraint

The regular constraint enforces that a sequence of n variables forms a word accepted by a finite automaton (NFA or DFA) with states Q and alphabet Σ.

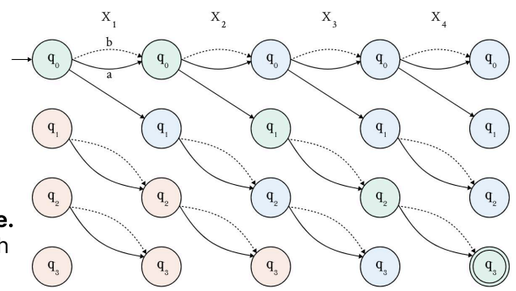
NFA (4 States) ⟷ Blowup Ratio: 2 ⟷ **DFA (8 States)**



Both automata accept sequences where the third-to-last symbol is 'a'.

Propagating the Regular Constraint

- Layered graph, with one layer per position.
- Edges are transitions allowed by current domains.
- Nodes are kept iff they are **reachable** and **co-reachable.**
  - i.e. they lie on a valid path from start to accepting.



2. Research Questions

- Both representations achieve identical inference strength (*GAC*), but NFAs can be far more compact.
- Explanations are generated greedily for speed, not minimality.

RQ1 How does the representation (NFA vs DFA) affect propagation efficiency in LCG?

RQ2 How does the representation change the quality of learned clauses in LCG?

RQ3 How far are the greedy explanations from the minimal (optimal) explanations?

3. Methodology

Instance Generation
Nonograms, Polyominoes, Regex

↓

Stratified Instance Classification
Blowup Ratio (DFA/NFA): <2x | 2-10x | >10x

↓

Instance Selection
Equal Sampling: 47 Per Bin (141 Total)

↓

Evaluation Pipeline
NFA, DFA, Decomposition Solvers (600s limit)

4. Theoretical Analysis

Propagation Complexity - Worst Case

DFA $\mathcal{O}(n|Q_{DFA}||\Sigma|)$

NFA $\mathcal{O}(n|Q_{NFA}|^2|\Sigma|)$

Trade-off: NFA is quadratic in a small state count, DFA is linear in a potentially exponential one.

NFA vs DFA Explanation Differences

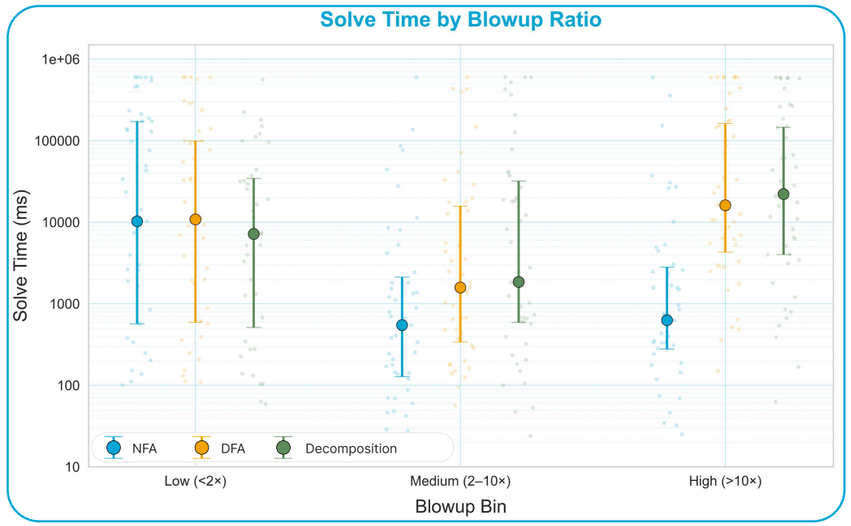
DFA: Follows a single chain through the layered graph.

NFA: Spans multiple parallel routes, but compactness of representation potentially offsets this.

5. Results

RQ1

Low Blowup	Med Blowup	High Blowup
NFA ≈ DFA	NFA	NFA
Within ~10%	*~3x Faster*	*~22x Faster*



RQ2 **Learned Clause Quality:** All metrics identical between NFA and DFA across all instances (nogood count, nogood length, LBD).

RQ3 **Greedy vs Minimal Explanations:** Greedy within ~1% overall, ~3-4% on nonograms, exact on regex. Polyominoes results uninterpretable.

6. Conclusion

NFAs should be the default representation for the regular constraint in LCG.

- ~22x faster than DFA at high blowup, never clearly worse in any regime.
- Clause quality matches DFA, and greedy explanations are near-optimal.

7. Future Work

- **Exploit non-determinism:** Current NFA propagators mimic DFA structure. Set-based transitions may improve propagation and explanations.
- **NFA-specific optimisations:** Extend runtime improvements (bit-vector, incremental propagation) from DFAs to NFAs.