Minimizing Speculation Overhead in a Parallel Recognizer for Regular Texts (PPoPP 2025, Mar 1-5, Las Vegas, NV, USA)

Angelo Borsotti [1], Luca Breveglieri [1], Stefano Crespi Reghizzi [1,2], Angelo Morzenti [1]

[1] Dipartimento di Elettronica Informazione e Bioingegneria (DEIB), Politecnico di Milano, Milano, Italy
[2] Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni (IEIIT), CNR, Milano, Italy

Introduction – context and challenge

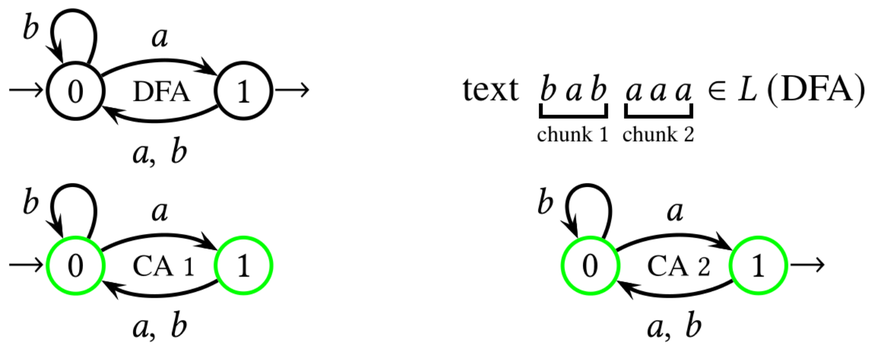
Parallel recognition of regular texts and its options

- several applications recognize texts by means of finite-state automata (FA – DFA/NFA)
- FAs may be designed directly, or derived from REs and other language description types
- recognition can be parallelized for speed on FPGA, SIMD, GPU and multi-core systems

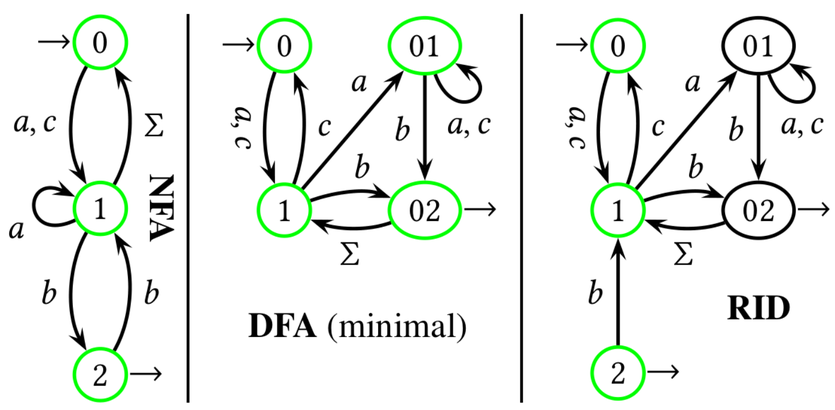
Classical speculative data-parallel algorithm (CSDPA)

- the text is split into *chunks* to be analyzed in parallel by as many *chunk automata* (CA)
- each CA is an instance of the recognizer DFA, where *all the states are taken as initial*
- each CA (but the first) has to *speculatively start a chunk run from every state* – see [2]

CSDPA challenge is to curb speculation overhead by reducing CA state number



CA 2 has to speculatively start in **every state** a run for its chunk
after all runs are done, consecutive state-matching runs are joined
text is accepted if join reaches the final state of the original DFA



NFA, equivalent powerset minimal DFA and RID

1. Reducing the number of initial states of the CA – Reduced Interface Device (RID)

- for most applications, the language is defined with a regular expression (RE) or an FA, often an NFA
- REs can be turned into NFAs with well-known algorithms (Gluskov-McNaughton-Yamada and others)
- RID is built from NFA by applying the well-known powerset construction to each NFA state separately
- identical states produced by each powerset construction are shared, *only the NFA states are initial*
- RID is deterministic, but *it may have two or more initial states*, which are *exactly those of the NFA*

RID is a so-called multi-entry DFA, deterministic except for having ≥ 1 initial states

2. How executing CSDPA with DFA or NFA or RID compares – states and transitions

What changes when CSDPA uses as its CA the minimal DFA, an NFA or our novel RID

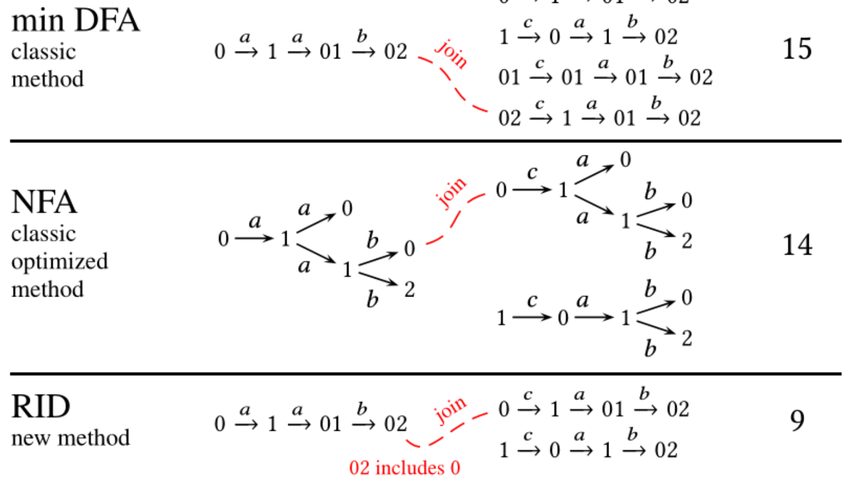
- minimal DFA has as few states as possible with determinism, and *one run per state*
- NFA often has *fewer states than the minimal DFA*, but two or more runs per state
- RID combines DFA and NFA: has *as few states as NFA*, and *only one run per state*

A two-chunk text analized with min. DFA, NFA and RID – number of state transitions

- minimal DFA performs *worst* with 15 transitions, since it has 4 states and 5 runs (chunks 1 and 2)
- NFA performs *better*, yet it still loses with 14 transitions, since its 3 runs split by nondeterminism
- RID performs *best* with 9 transitions, since it has 3 initial states (as NFA) but its runs do not split
- the join of consecutive matching runs is a fast operation ($\leq 1\%$ of total time) and does not matter

CSDPA can be implemented on multi-core computers – one RID-thread for each chunk

RID used as CA can curb speculation overhead – few states and one run per state

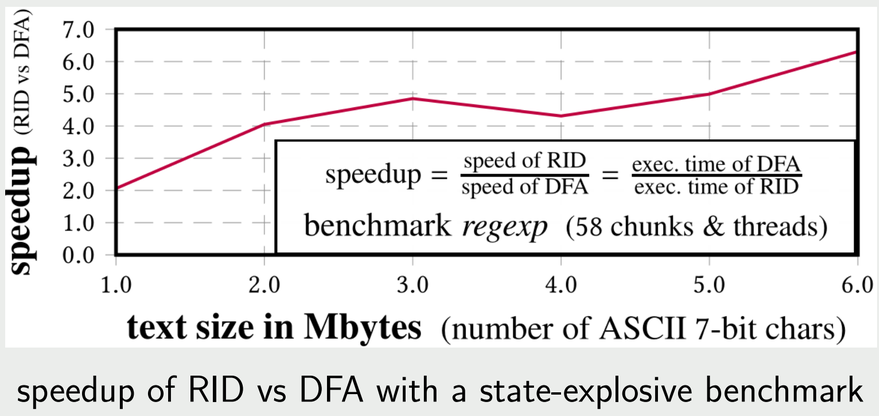


text a a b c a b – transitions with DFA, NFA and RID
chunk 1 chunk 2

3. Benchmarks and performance evaluation – speedup (on multi-core) and ratios of transition numbers for CSDPA with RID vs DFA & NFA

name	n. of NFAs	n. of states	max text length
Ondrik*	1084	2490 (avg)	none
large collection of NFAs that have different purposes			
bigdata	1	5	13 Mbyte
simple synthetic NFA from a short regular expression			
regexp	a series	$k+1 \geq 1$	6 Mbyte
series of synthetic NFAs, the DFAs of which grow exponentially, from the well-known REs $(a \mid b)^* a\,(a \mid b)^k$ with parameter $k \geq 0$			
bible*	1	16	4 Mbyte
HTML manuscript of The Holy Bible			
fasta*	1	29	765 Kbyte
biometric data consisting of various DNA sequences			
traffic*	1	101	11 Mbyte
system log file of network traffic records			

benchmark		speedup (RID vs FA) execution time ratio DFA/RID	speedup (RID vs FA) execution time ratio NFA/RID	transition ratio DFA/RID	transition ratio NFA/RID	text (Mbyte)
bigdata	even	1.01	73.24	1.00	1.99	13.10
regexp	winning	6.31	56.56	126.99	14.68	6.00
bible	winning	3.07	84.23	8.73	4.19	4.00
fasta	even	0.94	38.85	1.00	26.26	0.76
traffic	even	0.97	109.56	1.00	1.74	11.20



performance evaluation benchmarks — server with 64 cores and 58 RID-threads (= chunks) — speedup of RID vs DFA with a state-explosive benchmark

Main claims demonstrated by artifact & benchmarks

- RID is faster than DFA for the winner benchmarks, equally fast for the even ones (within 10%), and is always much faster than NFA
- optimal speedup occurs when an NFA is smaller than its minimal DFA
- a large collection of big NFAs meets condition NFA $\lll$ DFA (Ondrik)
- the time cost to construct RID (from an RE or an NFA) is moderate
- the full paper is posted on arXiv with more theory and evaluation [1]

Conclusions

RID is compatible with most other optimizations for FAs and it should be a useful addition for future parallel implementations of finite-state machines

References

[1] Angelo Borsotti et al. *Minimizing speculation overhead in a parallel recognizer for regular texts*. 2024. DOI: https://doi.org/10.48550/arXiv.2412.14975. arXiv: 2412.14975 [cs.DC]. URL: https://arxiv.org/abs/2412.14975.

[2] J. Holub and S. Stekr. "On parallel implementations of deterministic finite automata". In: *CIAA*. 2009. DOI: 10.1007/978-3-642-02979-0_9. URL: https://doi.org/10.1007/978-3-642-02979-0_9.

Artifact

- free SW tool to build DFA/NFA/RID (from RE) and run CSDPA with all FAs
- coded in Java and available on https://zenodo.org/records/14219357