Weak Factor Automata: Comparing (Failure) Oracles and Storacles

Loek Cleophas, Derrick G. Kourie, and Bruce W. Watson

FASTAR Research Group, University of Pretoria and Stellenbosch University, South Africa

{loek,derrick,bruce}@fastar.org

http://www.fastar.org

Prague Stringology Conference, 2–4 September 2013

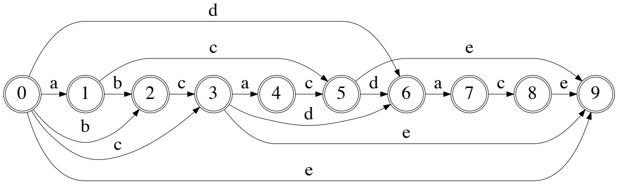
(Weak) factor automata

- Factor automaton (DAWG)
  - Accepts factors of keyword
  - Used for efficient backward pattern matching
  - Used as index
- Weak factor automata
  - (Small) over-approximation
    ... to save space
  - OK for pattern matching
  - May be OK for indexing

Contributions

- New weak factor automata constructions
  - Based on factor oracle and factor storacle
  - Using failure transitions
  - Failure factor oracle
  - Failure factor storacle
- Empirical size comparison
  - On generated strings of lengths $4 - 9$
  - On English word list (lengths $4 - 28$)

Factor oracle



- Small over-approximation—e.g. *bce*, *cace*
- $m+1$ states, m to $2m-1$ transitions
- Acylic
- Homogeneous
- $O(m)$ construction (*Allauzen, Crochemore & Raffinot 1999*)
- $O(m^2)$ construction (*Cleophas, Zwaan & Watson 2003/2005*)
  - Conceptually simpler, some properties obvious

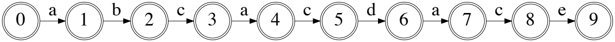
fastar RESEARCH GROUP

Factor oracle construction algorithm

for i from 0 to m **do**
    Create a new final state i
for i from 0 to $m-1$ **do**
    Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
    Let the longest path from state 0 that spells a prefix of $p_i ... p_m$ end in state j and spell out $p_i ... p_k$ $(i-1 \leq k \leq m)$
    if $k \neq m$ **then**
        Build a new transition $j \stackrel{p_{k+1}}{\rightarrow} k+1$

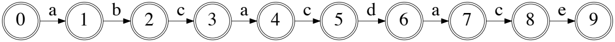
Factor oracle construction algorithm

for i from 0 to m **do**
    Create a new final state i
for i from 0 to $m-1$ **do**
    Create a new transition from i to $i+1$ on symbol p_{i+1}



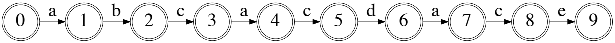
Factor oracle construction algorithm

for i from 0 to m **do**
    Create a new final state i
for i from 0 to $m-1$ **do**
    Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
    Let the longest path from state 0 that spells a prefix of $p_i \ldots p_m$ end in state j and spell out $p_i \ldots p_k$ $(i-1 \leq k \leq m)$



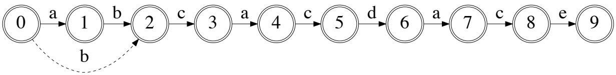
Factor oracle construction algorithm

for i from 0 to m **do**
  Create a new final state i
for i from 0 to $m-1$ **do**
  Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
  Let the longest path from state 0 that spells a prefix of $p_i \ldots p_m$ end in state j and spell out $p_i \ldots p_k$ $(i-1 \leq k \leq m)$
  if $k \neq m$ **then**
    Build a new transition $j \stackrel{p_{k+1}}{\rightarrow} k+1$



Factor oracle construction algorithm

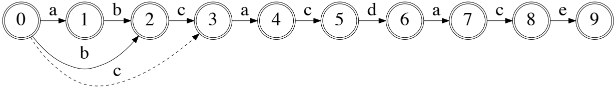
for i from 0 to m **do**
  Create a new final state i
for i from 0 to $m-1$ **do**
  Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
  Let the longest path from state 0 that spells a prefix of $p_i \ldots p_m$ end in state j and spell out $p_i \ldots p_k$ $(i-1 \le k \le m)$
  if $k \ne m$ **then**
    Build a new transition $j \stackrel{p_{k+1}}{\rightarrow} k+1$



Suffix bcacdace

Factor oracle construction algorithm

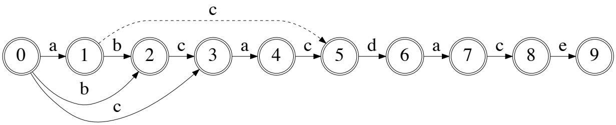
for i from 0 to m **do**
    Create a new final state i
for i from 0 to $m-1$ **do**
    Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
    Let the longest path from state 0 that spells a prefix of $p_i \ldots p_m$ end in state j and spell out $p_i \ldots p_k$ $(i-1 \leq k \leq m)$
    if $k \neq m$ **then**
        Build a new transition $j \stackrel{p_{k+1}}{\rightarrow} k+1$



Suffix cacdace

Factor oracle construction algorithm

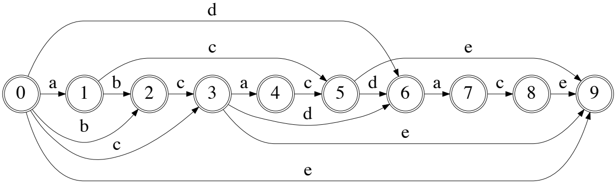
for i from 0 to m **do**
  Create a new final state i
for i from 0 to $m-1$ **do**
  Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
  Let the longest path from state 0 that spells a prefix of $p_i \ldots p_m$ end in state j and spell out $p_i \ldots p_k$ $(i-1 \le k \le m)$
  if $k \neq m$ **then**
    Build a new transition $j \overset{p_{k+1}}{\rightarrow} k+1$



Suffix acdace

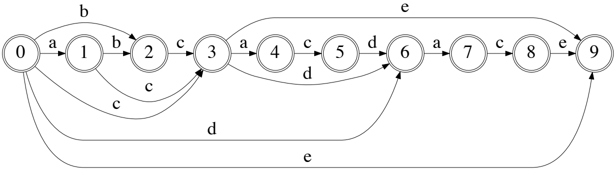
Factor oracle

... etc. leads to



Factor storacle

- Modification to $O(m^2)$ FO construction
- shortest forward **t**ransition **oracle**
  - ... keeping it homogeneous
  - Accidental...
  - Example smaller than FO



- Not smaller than FO in general (*Cleophas & Watson 2012*)

  ... usually *slightly larger*
- Conjecture m to $3m$ transitions

fastar
RESEARCH GROUP

Factor storacle construction algorithm

for i from 0 to m **do**
  Create a new final state i
for i from 0 to $m-1$ **do**
  Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
  Let the longest path from state 0 that spells a prefix of $p_i ... p_m$ end in state j and spell out $p_i ... p_k$ $(i-1 \leq k \leq m)$
  while $k \neq m$ **do**
    Let the first state from state j onward that has an incoming transition on p_{k+1} be state l $(j < l <= k+1)$
    Build a new transition $j \stackrel{p_{k+1}}{\rightarrow} l$
    Let the longest path from state 0 that spells a prefix of $p_i ... p_m$ end in state j and spell out $p_i ... p_k$ $(i-1 \leq k \leq m)$

Factor storacle construction algorithm

```
for i from 0 to m do
    Create a new final state i
for i from 0 to m − 1 do
    Create a new transition from i to i + 1 on symbol p_{i+1}
```

Factor storacle construction algorithm

for i from 0 to m **do**
    Create a new final state i
for i from 0 to $m-1$ **do**
    Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
    Let the longest path from state 0 that spells a prefix of $p_i \ldots p_m$ end in state j and spell out $p_i \ldots p_k$ $(i-1 \leq k \leq m)$

        Let the first state from state j onward that has an incoming transition on p_{k+1} be state l $(j < l <= k+1)$
        Build a new transition $j \overset{p_{k+1}}{\rightarrow} l$

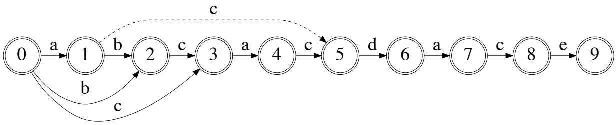
fastar RESEARCH GROUP

Factor storacle construction algorithm

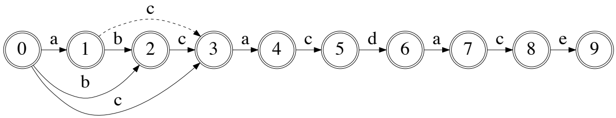
for i from 0 to m **do**
    Create a new final state i
for i from 0 to $m-1$ **do**
    Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
    Let the longest path from state 0 that spells a prefix of $p_i \ldots p_m$ end in state j and spell out $p_i \ldots p_k$ $(i-1 \leq k \leq m)$
    while $k \neq m$ **do**
        Let the first state from state j onward that has an incoming transition on p_{k+1} be state l $(j < l <= k+1)$
        Build a new transition $j \overset{p_{k+1}}{\rightarrow} l$
        Let the longest path from state 0 that spells a prefix of $p_i \ldots p_m$ end in state j and spell out $p_i \ldots p_k$ $(i-1 \leq k \leq m)$

Factor storacle construction example

Recall factor oracle case:

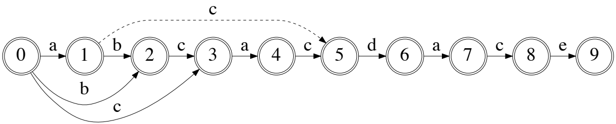


Factor storacle case:

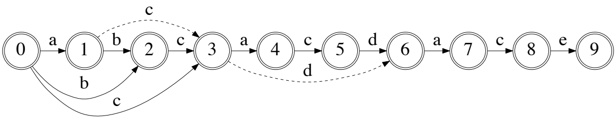


Factor storacle construction example

Recall factor oracle case:



Factor storacle case:



Failure transitions

- Allow failure transitions in addition to symbol ones
- Save space, but more transition use...
- Not new
  - Aho-Corasick
  - Generalized by *Crochemore & Hancart 1997*
- First general *DFA* $\rightarrow$ *FDFA* algorithm by *Kourie et al. 2012*
  - Intermediate lattice construction
    ... keeping state set fixed
- *Björklund et al. 2013*
  - Complexity...
    ... even if keeping state set fixed
  - Algorithm to reach $\frac{2}{3}$ of optimal savings
- Both *ex post facto*...

Our idea

- Introduce failure transitions *during construction*
- We call the resulting automata *Failure Factor (St)Oracles*
- Complexity as for non-failure cases
- Idea: instead of constructing $j \xrightarrow{a} k+1$ ...

  ... construct $j \xrightarrow{fail} k$, from which $k \xrightarrow{a} k+1$ exists
- Potential problem...

fastar RESEARCH GROUP

Our idea

- Introduce failure transitions *during construction*
- We call the resulting automata *Failure Factor (St)Oracles*
- Complexity as for non-failure cases
- Idea: instead of constructing $j \stackrel{a}{\rightarrow} k+1$ ...

  ... construct $j \stackrel{fail}{\rightarrow} k$, from which $k \stackrel{a}{\rightarrow} k+1$ exists
- Potential problem...
- Using sequence of existing (failure, symbol) transitions may end up in $j > k$

Potential problem...

- Using sequence of existing (failure, symbol) transitions may end up in $j > k$

  ... potential for *backward* failure transition

  ... hence for *cycle*

  ... hence for *failure cycle*

  ... hence for *divergent failure cycle*

  ... leading to live-lock in construction or use of automaton
- Solution: do not construct backward *failure* transition

  ... instead create non-forward *symbol* transition

  ... still potential for cycle (but manageable)

Failure factor oracle construction algorithm

for i from 0 to m **do**
  Create a new final state i
for i from 0 to $m-1$ **do**
  Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
  Let the longest recognized prefix of $p_i \ldots p_m$ be recognized in state j' and spell out $p_i \ldots p_k$ $(i-1 \leq k \leq m)$, and let the longest failure transition path from j' end in state j
  if $k \neq m$ **then**
    if $k > j$ **then**
      Build a new failure transition $j \overset{fail}{\rightarrow} k$
    else
      Build a new symbol transition $j \overset{p_{k+1}}{\rightarrow} k+1$

Failure factor oracle construction algorithm

```
for i from 0 to m do
    Create a new final state i
for i from 0 to m − 1 do
    Create a new transition from i to i + 1 on symbol p_{i+1}
```

Failure factor oracle construction algorithm

for i from 0 to m **do**
    Create a new final state i
for i from 0 to $m-1$ **do**
    Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
    Let the longest recognized prefix of $p_i \ldots p_m$ be recognized in state j' and spell out $p_i \ldots p_k$ $(i-1 \leq k \leq m)$, and let the longest failure transition path from j' end in state j

Failure factor oracle construction algorithm

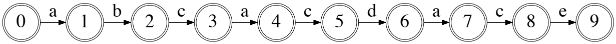
for i from 0 to m **do**
    Create a new final state i
for i from 0 to $m-1$ **do**
    Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
    Let the longest recognized prefix of $p_i \ldots p_m$ be recognized in state j' and spell out $p_i \ldots p_k$ $(i-1 \leq k \leq m)$, and let the longest failure transition path from j' end in state j
    if $k \neq m$ **then**
        if $k > j$ **then**
            Build a new failure transition $j \stackrel{fail}{\rightarrow} k$
        else
            Build a new symbol transition $j \stackrel{p_{k+1}}{\rightarrow} k+1$

Failure factor oracle construction algorithm

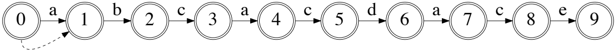
for i from 0 to m **do**
    Create a new final state i
for i from 0 to $m-1$ **do**
    Create a new transition from i to $i+1$ on symbol p_{i+1}
for i from 2 to m **do**
    Let the longest recognized prefix of $p_i \ldots p_m$ be recognized in state j' and spell out $p_i \ldots p_k$ $(i-1 \leq k \leq m)$, and let the longest failure transition path from j' end in state j
    if $k \neq m$ **then**
        if $k > j$ **then**
            Build a new failure transition $j \stackrel{fail}{\rightarrow} k$
        else
            Build a new symbol transition $j \stackrel{p_{k+1}}{\rightarrow} k+1$

Failure factor storacle construction algorithm similar

Failure factor oracle construction example

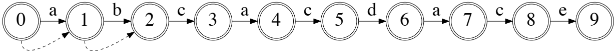


Failure factor oracle construction example



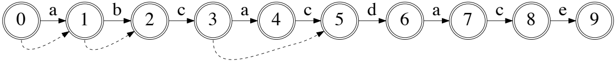
Suffix bcacdace

Failure factor oracle construction example



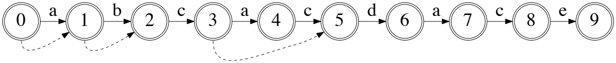
Suffix cacdace

Failure factor oracle construction example



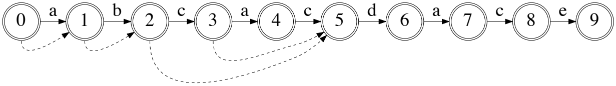
Suffix acdace

Failure factor oracle construction example



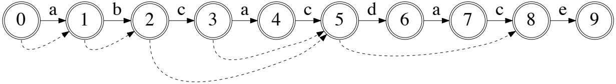
Suffix acdace
Suffix cdace

Failure factor oracle construction example



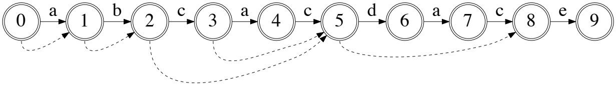
Suffix dace

Failure factor oracle construction example



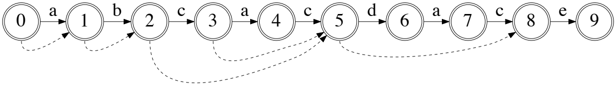
Suffix ace

Failure factor oracle construction example



Suffix ace
Suffix ce

Failure factor oracle construction example



Suffix ace

Suffix ce

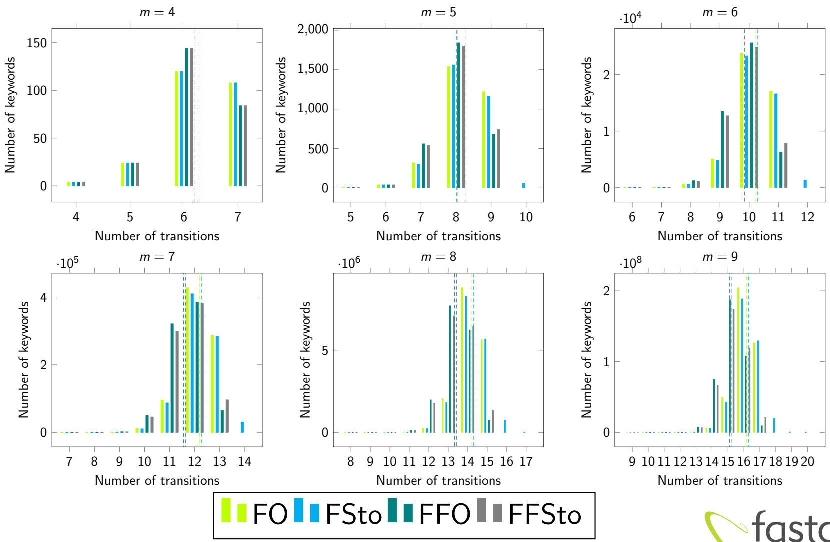
Suffix e

Empirical results

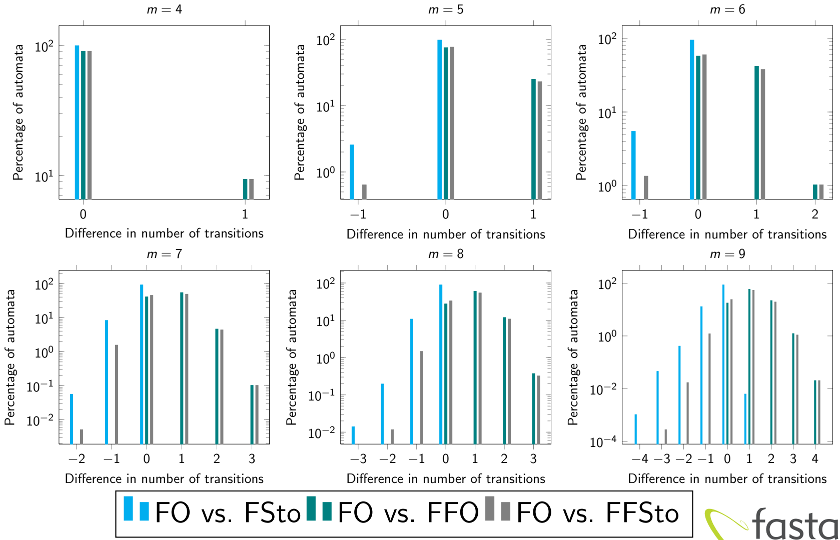
Two data sets

- Generated strings: all strings of length $m \in [4, 9]$ over alphabet of size m
- English words

Generated strings—number of transitions

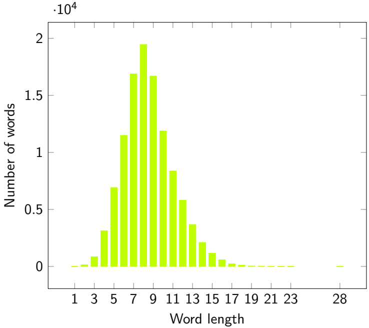


Generated strings—difference in #transitions of FO vs. ...



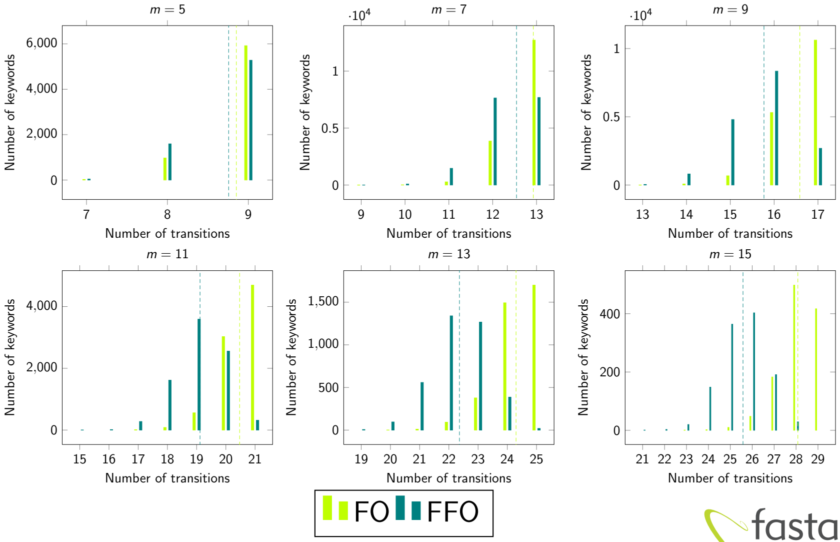
Empirical results on English words

- English word list from `http://www.sil.org/linguistics/wordlists/english`.

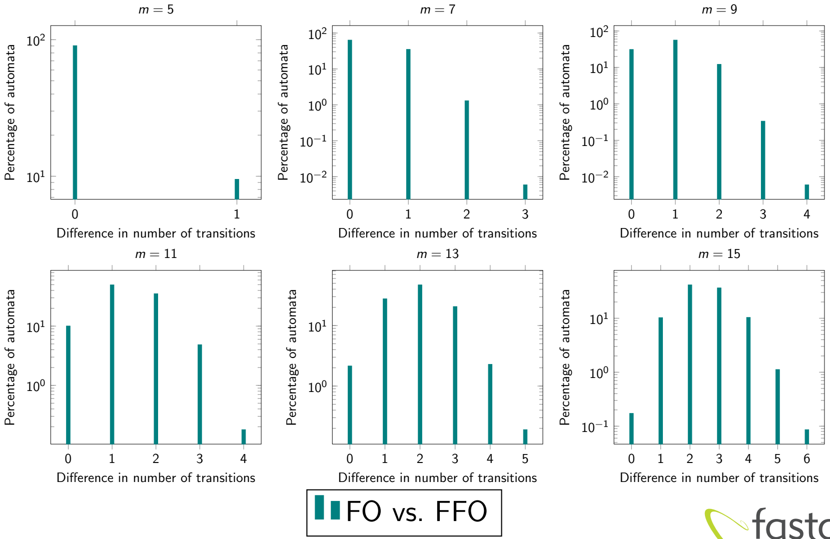


... disregarding words p with $|p| < 4$

English words—number of transitions



English words—difference in #transitions of FO vs. FFO



Concluding remarks

- Failure versions save ca. 2-10% on #transitions

  ... possibly more on space
- Open questions
  - Upper bounds on number of transitions
  - Languages
  - Comparison to general super automata
  - Comparison to general FDFA construction algorithm
- Performance when using automata

  ... recent work: FFO for DNA strings of lengths $4 - 2048$
  - Savings of $8 - 10\%$ for lengths $16 - 2048$
  - Also rudimentary processing; runtime increases $34 - 88\%$

fastar RESEARCH GROUP

References

- Allauzen, Crochemore & Raffinot, *Factor oracle: a new structure for pattern matching*, SOFSEM 1999.
- Björklund, Björklund & Zechner, *Compact representation of finite automata with failure transitions*, Umea Univ. TR 2013/011.
- Cleophas & Watson, *On Factor Storacles: an Alternative to Factor Oracles?*, Festschrift for Bořivoj Melichar, 2012.
- Cleophas, Zwaan & Watson, *Constructing Factor Oracles*, PSC 2003 and JALC special issue 2005.
- Crochemore & Hancart, *Automata for matching patterns*, pp. 399–462 in Handbook of Formal Languages, 1997.
- Kourie, Watson, Cleophas & Venter, *Failure Deterministic Finite Automata*, PSC 2012.

fastar
RESEARCH GROUP