An Improved Algorithm to Accelerate Regular Expression Evaluation

Michela Becchi and Patrick Crowley

ANCS 2007
Context

- Regular expression matching is a critical operation in networking
  - Intrusion detection
  - Context based billing
  - Peer-to-peer traffic detection and prioritization
  - Application level filtering

- Challenge: perform regular expression matching at line rate, given data-sets of hundreds (or thousands) of patterns
  - Processing time
  - Memory requirement (occupancy and bandwidth)
Two algorithmic solutions

- Non deterministic finite automata (NFAs)
  - High time complexity/memory bandwidth requirements
  - Compact representation
- Deterministic finite automata (DFAs)
  - Low time complexity
  - Potentially high storage requirement

Multiple implementation approaches

- Custom hardware [Kumar 2006]

Problem: given a DFA, find a representation

1. compact
2. allowing an acceptable bound of memory bandwidth requirement/processing time
Observation:
» DFAs from practical datasets have redundancy in state transitions

Idea:
» *default transitions*: non-consuming transitions

Implication:
» Traversal time / memory bandwidth requirement dependent upon maximum default path length
Background – $D^2$FA construction

RegEx: $ab^+c^+$, $cd^+$ and $bd^+e$

Remaining transitions

Space reduction graph
Background – $D^2$FA construction

RegEx: $ab^+c^+$, $cd^+$ and $bd^+e$

Remaining transitions from 1-8

Space reduction graph

Diameter bound = 4 \Rightarrow removed transitions = 33
Background – D²FA construction

RegEx: $ab^+c^+$, $cd^+$ and $bd^+e$

Remaining transitions from 1-8

Space reduction graph

D²FA

Diameter bound = 4 ⇒ removed transitions = 33
Background – D²FA construction

RegEx: $ab^+c^+, cd^+$ and $bd^+e$

Space reduction graph

D²FA

Traversal time=$O((D/2+1)N)$
Time complexity=$O(n^2\log n)$
Space complexity=$O(n^2)$
Transition redundancy: why?

RegEx: \(ab^+c^+, cd^+ \) and \(bd^+e\)

- **Forward** transitions:
  - Matches
  - State specific

- **Backward** transitions:
  - Mismatch
  - Shared by multiple states

**Idea:**
- Introduce *state depth*: minimum distance from entry state
- Orient default transitions only backwards (towards decreasing depth)

**Pros:**
- Traversal time \(O(2N)\) independent of the maximum default path length
- Generality: no need of diameter bound parameter

**Cons:**
- Possible compression loss
Our scheme

RegEx: $ab^+c^+, cd^+ \text{ and } bd^+e$

Observations:
- Maximum spanning tree on oriented graph: Edmonds and Chu solutions
  - 2 steps: edge selection and cycle resolution
- No cycles:
  - Space reduction graph not necessary
  - Simple breath-first traversal algorithm

Traversal time=$O(2N)$
Time complexity=$O(n^2)$
Space complexity=$O(n)$
Discussion

- Generalization (Jon Turner’s observation)
  - Allowing default transitions only from depth $d$ to depth $\leq d-k$, w/ $k \geq 1$, leads to worst case traversal time $O\left(\frac{k+1}{k}N\right)$
    - Time and space complexity of the construction algorithm still $O(n^2)$ and $O(n)$
    - Examples:
      - $k=1 \Rightarrow \text{traversal } O(2N)$
      - $k=2 \Rightarrow \text{traversal } O(1.5N)$
      - $k=3 \Rightarrow \text{traversal } O(1.33N)$
      - $k=4 \Rightarrow \text{traversal } O(1.25N)$

- Compression
  - $D^2FA$:
    - Constraint: diameter bound
    - Heuristic
  - Our algorithm:
    - Constraint: orientation (may be not a problem for RegEx originated DFAs)
    - Optimal solution
Discussion (cont’d)

- Default transitions and depth computation through breath-first traversal
  » Default transitions can be computed during subset construction, that is, at DFA creation time.

- $D^2$FA space complexity can be an issue for big DFAs
  » $O(n^2)$: space reduction graph
  » Using adjacency list 17B/edge
    ```
    struct wgedge {
        vertex l,r;       // endpoints of the edge
        weight wt;        // edge weight
        edge lnext;       // link to next edge incident to l
        edge rnext;       // link to next edge incident to r
    } *edges;
    ```
  » Fully connected graph w/ ~11K nodes will require 1GB storage
  » Possible solutions: partial graphs based on weight
    - Multiple scans
    - Effect on algorithm’s execution time
Discussion (cont’d)

- Traversal locality
  - DFA traversal exhibits locality
  - Average traffic tends to mismatch
  - States at low depths tend to be traversed more
  - Backward default transition reiterate the traversal of likely states
Alphabet reduction

- **Observation:**
  - Some symbols are treated in the same way over the whole DFA \[ \delta(s, c_i) = \delta(s, c_j) \] for each state \( s \in \text{DFA} \)
  - Example:
    - Ignore case
    - \n, \r
    - unused characters

- **Idea:**
  - Group characters into classes
  - Mapping filter

- **Algorithm:**
  - Sequence of clustering operations
  - Breath-first traversal w/ \( O(n^2) \) complexity
  - Applicable at DFA creation time
Evaluation: rule-sets

<table>
<thead>
<tr>
<th>Data-set</th>
<th>ASCII length range</th>
<th>% RegEx w/ wild-cards (*),+</th>
<th>% RegEx w/ char ranges $\geq$ 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snort24</td>
<td>6..70</td>
<td>37.5</td>
<td>50</td>
</tr>
<tr>
<td>Snort34</td>
<td>15..99</td>
<td>38.2</td>
<td>32.4</td>
</tr>
<tr>
<td>Snort31</td>
<td>16..120</td>
<td>41.9</td>
<td>93.5</td>
</tr>
<tr>
<td>Cisco11</td>
<td>9..13</td>
<td>90.9</td>
<td>9.1</td>
</tr>
<tr>
<td>Cisco43</td>
<td>15..73</td>
<td>32.6</td>
<td>27.9</td>
</tr>
<tr>
<td>Cisco612</td>
<td>3..50</td>
<td>0</td>
<td>1.6</td>
</tr>
<tr>
<td>Bro217</td>
<td>5..76</td>
<td>1.4</td>
<td>13.4</td>
</tr>
</tbody>
</table>
## Evaluation - compression

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Original DFA</th>
<th>D²FA algorithm</th>
<th>Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of states</td>
<td>% duplicates</td>
<td>max def. length</td>
</tr>
<tr>
<td>Snort24</td>
<td>13886</td>
<td>98.97</td>
<td>89.59 98.48 98.91 98.92 98.92</td>
</tr>
<tr>
<td>Snort34</td>
<td>13825</td>
<td>98.91</td>
<td>88.33 98.48 98.85 98.86 98.86</td>
</tr>
<tr>
<td>Snort31</td>
<td>20052</td>
<td>98.93</td>
<td>74.42 97.18 98.42 98.6 98.63</td>
</tr>
<tr>
<td>Cisco11</td>
<td>24011</td>
<td>97.45</td>
<td>86.73 97.08 97.37 97.38 97.38</td>
</tr>
<tr>
<td>Cisco43</td>
<td>20320</td>
<td>99.06</td>
<td>90.16 98.46 99 99.05 99.05</td>
</tr>
<tr>
<td>Cisco612</td>
<td>11309</td>
<td>99.5</td>
<td>79.3 97.46 98.93 99.18 99.25</td>
</tr>
<tr>
<td>Bro217</td>
<td>6533</td>
<td>99.57</td>
<td>76.49 97.9 99.07 99.4 99.41</td>
</tr>
</tbody>
</table>

The table above shows the compression results for different datasets using the D²FA algorithm and our algorithm. The columns represent the compression ratio as a function of the diameter bound (DB) and the maximum definition length for each algorithm.
Evaluation – number of transitions

- Distinct transitions
- Our algorithm
- D2FA, DB=2
- D2FA, DB=∞

Rule-set

- Snort24
- Snort34
- Snort31
- Cisco11
- Cisco43
- Cisco612
- Bro217

Number of transitions

- Snort24: x8
- Snort34: x8
- Snort31: x16
- Cisco11: x4
- Cisco43: x9.5
- Cisco612: x23
- Bro217: x35

Michela Becchi - 1/9/2008
## Alphabet reduction’s effect

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of nodes</th>
<th>alphabet size</th>
<th>D²FA, DB=2</th>
<th>D²FA, DB=∞</th>
<th>Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>compression %</td>
<td>transitions after AR</td>
<td>compression %</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BAR</td>
<td>AAR</td>
<td>BAR</td>
</tr>
<tr>
<td>Snort24</td>
<td>13886</td>
<td>46</td>
<td>89.59</td>
<td>97.87</td>
<td>75752</td>
</tr>
<tr>
<td>Snort34</td>
<td>13825</td>
<td>51</td>
<td>89.33</td>
<td>97.63</td>
<td>84046</td>
</tr>
<tr>
<td>Snort31</td>
<td>20052</td>
<td>53</td>
<td>74.42</td>
<td>94.48</td>
<td>283339</td>
</tr>
<tr>
<td>Cisco11</td>
<td>24011</td>
<td>38</td>
<td>86.73</td>
<td>97.74</td>
<td>138922</td>
</tr>
<tr>
<td>Cisco43</td>
<td>20320</td>
<td>65</td>
<td>90.16</td>
<td>97.09</td>
<td>151161</td>
</tr>
<tr>
<td>Cisco612</td>
<td>11309</td>
<td>115</td>
<td>79.3</td>
<td>90.46</td>
<td>276110</td>
</tr>
<tr>
<td>Bro217</td>
<td>6533</td>
<td>111</td>
<td>76.49</td>
<td>89.59</td>
<td>174035</td>
</tr>
</tbody>
</table>
Further decreasing the traversal time

<table>
<thead>
<tr>
<th>Rule-set</th>
<th># of states</th>
<th>% of duplicates</th>
<th>D²FA</th>
<th>Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DB=2</td>
<td>DB=∞</td>
<td>k=1</td>
<td>k=2</td>
</tr>
<tr>
<td>Bro217</td>
<td>6533</td>
<td>99.57</td>
<td>89.59</td>
<td>99.33</td>
</tr>
<tr>
<td>Cisco11</td>
<td>24011</td>
<td>97.45</td>
<td>89.33</td>
<td>96.63</td>
</tr>
<tr>
<td>Cisco43</td>
<td>20320</td>
<td>99.06</td>
<td>74.42</td>
<td>98.97</td>
</tr>
<tr>
<td>Cisco613</td>
<td>11309</td>
<td>99.5</td>
<td>86.73</td>
<td>99.09</td>
</tr>
<tr>
<td>Snort24</td>
<td>13886</td>
<td>98.97</td>
<td>90.16</td>
<td>98.71</td>
</tr>
<tr>
<td>Snort34</td>
<td>13825</td>
<td>98.91</td>
<td>79.3</td>
<td>98.69</td>
</tr>
</tbody>
</table>
Conclusion

- DFA exhibit transition redundancy exploitable through default transitions

- $D^2$FA: algorithm trading off compression w/ traversal time

- In this work, we propose generic algorithm:
  » With limited time and space complexity
  » Allowing $O(2N)$ traversal time (or less) when processing input text
  » Leading to compression level similar to (or better than) $D^2$FA
Thank you!

Questions?
Default transitions targets

Distribution of depth of the default transitions' targets for Cisco613 data-set
Default transitions targets (cont’d)

Distribution of depth of the default transitions' targets for Snort24 data-set.

% default transitions' targets

depth
Our algorithm

procedure default_transition (DFA dfa=(n, δ(states, Σ)), modifies set default);

  list queue; set depth[n];
  for state s ∈ states ⇒ depth[s]=n; default[s]=s; rof
  depth[0]=0; queue.push(0);
  while (!queue.empty()) ⇒
    state s = queue.pop();
    int saving=0;
    for char c ∈ Σ ⇒
      if (depth[δ(s,c)]=n) ⇒
        depth[δ(s,c)]= depth[s]+1; queue.push(δ(s,c));
      fi
    rof;

  for (state t ∈ states & depth[t]<depth[s]) ⇒
    int common:=# common transitions btw. s and t;
    if (common > 1 && (common>saving ||
      (common=saving && depth[t]<depth[default[s]])))
      saving:=common;
    default[s]=t;
  fi
  rof;
end while;
end;