Partial Determinization of Finite Automata

Denis Matoušek
LTA, 2015

Outline

- Motivation
- NFA vs. DFA in FPGA
- Hybrid FA
- System of Parallel Automatons Parts

Motivation

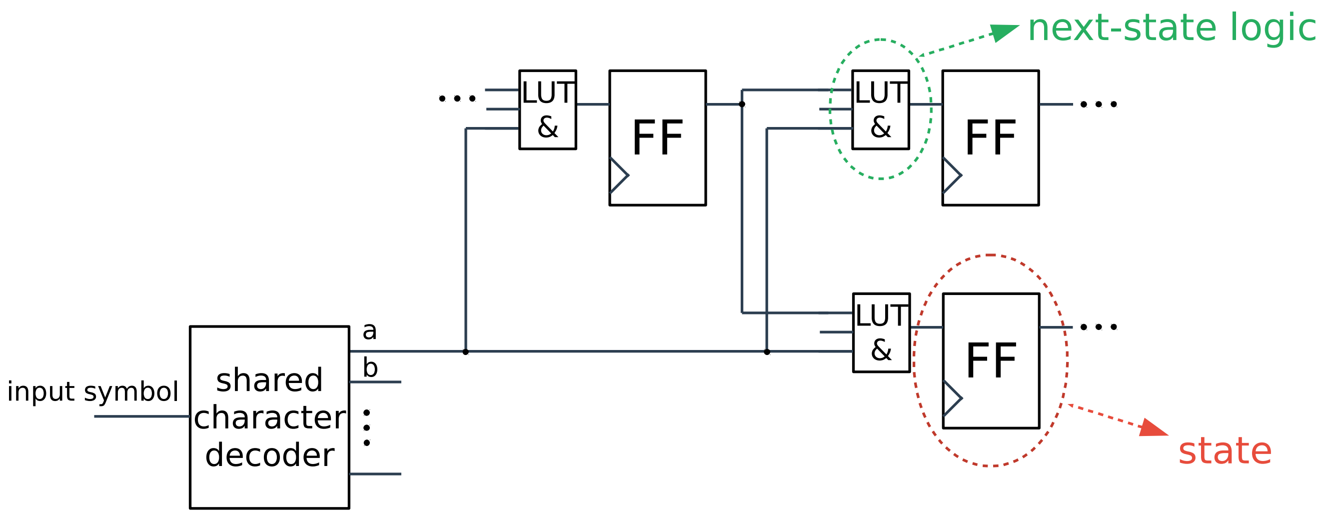
- **Network intrusion detection systems (NIDS) use rules described with regular expressions (RE)**
  - Significant state transition redundancy
- **Implementation of equivalent computational machine — finite automaton (FA) — in hardware is used to achieve high performance / throughput on high-speed network links through massive parallelism**
- **FPGA technology is used for ability to change the configuration (implement different FAs)**

Implementation in FPGA: DFA vs. NFA I

- **Tradeoff between DFA and NFA:**
  - NFA using FF registers (states) and LUTs (transitions)
  - DFA using BlockRAMs (storage of a hash table) and LUTs (computation of a hash function)
- **FA parameters of interest:**
  - Number of states
  - Maximum number of concurrently active states

Implementation in FPGA: DFA vs. NFA II

- **Tradeoff between DFA and NFA:**
  - **NFA using FF registers (states) and LUTs (transitions)**
  - DFA using BlockRAMs (storage of a hash table) and LUTs (computation of a hash function)

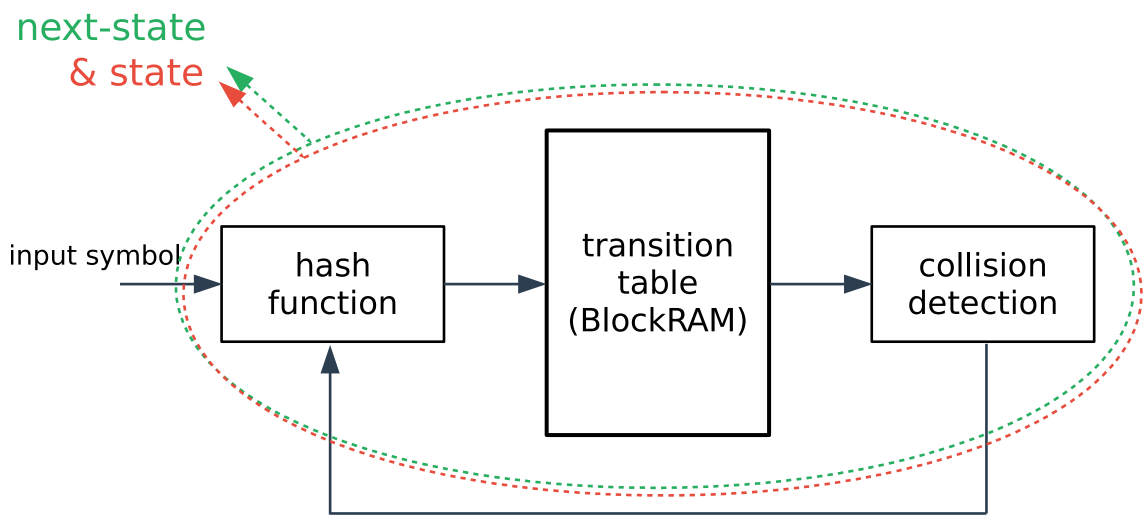


Implementation in FPGA: DFA vs. NFA III

- **Tradeoff between DFA and NFA:**
  - **NFA using FF registers (states) and LUTs (transitions)**
  - DFA using BlockRAMs (storage of a hash table) and LUTs (computation of a hash function)
- **FA parameters of interest:**
  - Number of states
  - **Maximum number of concurrently active states**

Implementation in FPGA: DFA vs. NFA IV

- **Tradeoff between DFA and NFA:**
  - NFA using FF registers (states) and LUTs (transitions)
  - **DFA using BlockRAMs (storage of a hash table) and LUTs (computation of a hash function)**



Implementation in FPGA: DFA vs. NFA V

- **Tradeoff between DFA and NFA:**
  - NFA using FF registers (states) and LUTs (transitions)
  - **DFA using BlockRAMs (storage of a hash table) and LUTs (computation of a hash function)**
- **FA parameters of interest:**
  - **Number of states**
  - Maximum number of concurrently active states

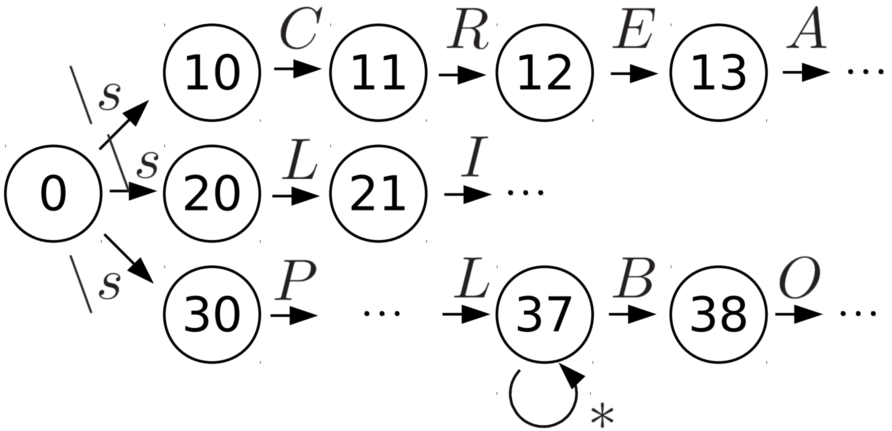
Typical Form of Rules Used in NIDS

- **Example:**
  - a part of IMAP ruleset of Snort (https://www.snort.org/) and corresponding NFA:

\sCREATE\s*\{

\sLIST\s[^\n]*?\s\{

\sPARTIAL.*BODY\[[^\]]{1024}



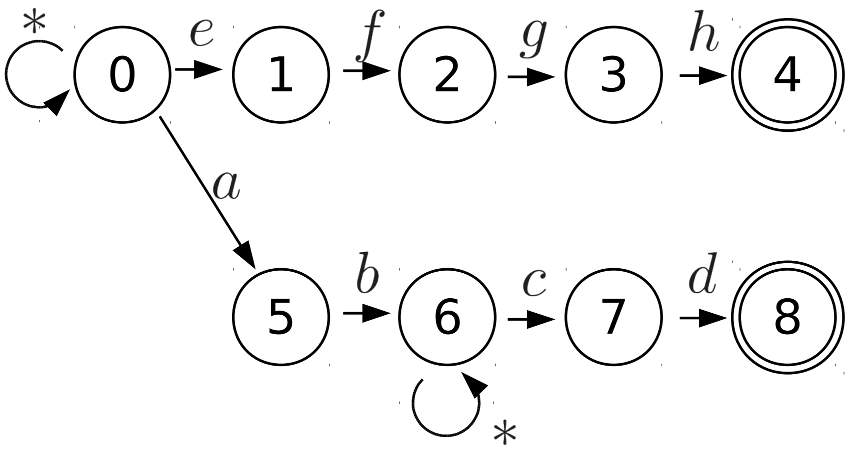
Problematic NFA Constructions I

- **Parts of REs that cause state explosion during NFA determinization**
- **Mainly "dot-star" constructions**
- **Example:**

  1. ab.*cd

  2. efgh

- Corresponding NFA:

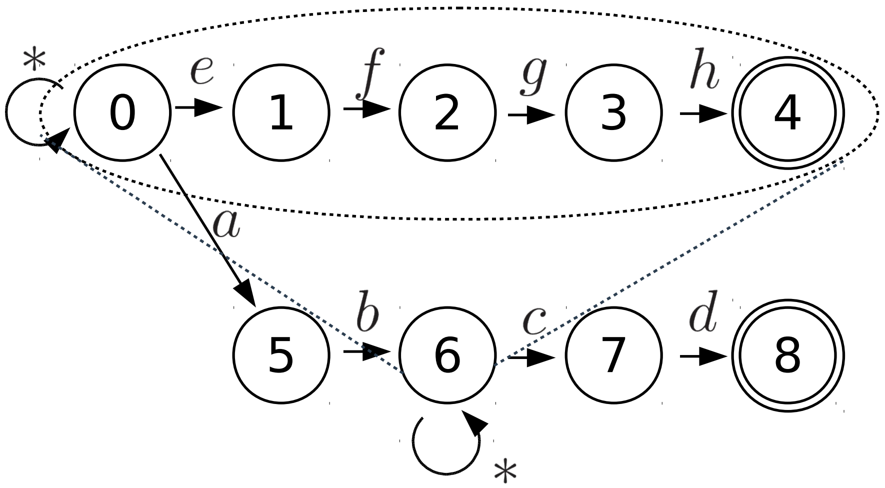


- Transitions to 0 omitted

Problematic NFA Constructions II

- **Parts of REs that cause state explosion during NFA determinization**
- **Mainly "dot-star" constructions**
- **Example:**
  1. ab.*cd
  2. efgh

- Corresponding NFA:



- Transitions to 0 omitted

Problematic NFA Constructions III

- **Parts of REs that cause state explosion during NFA determinization**
- **Mainly "dot-star" constructions**
- **Example:**

  1. ab.*cd

  2. efgh

  - Corresponding NFA (from [1]):

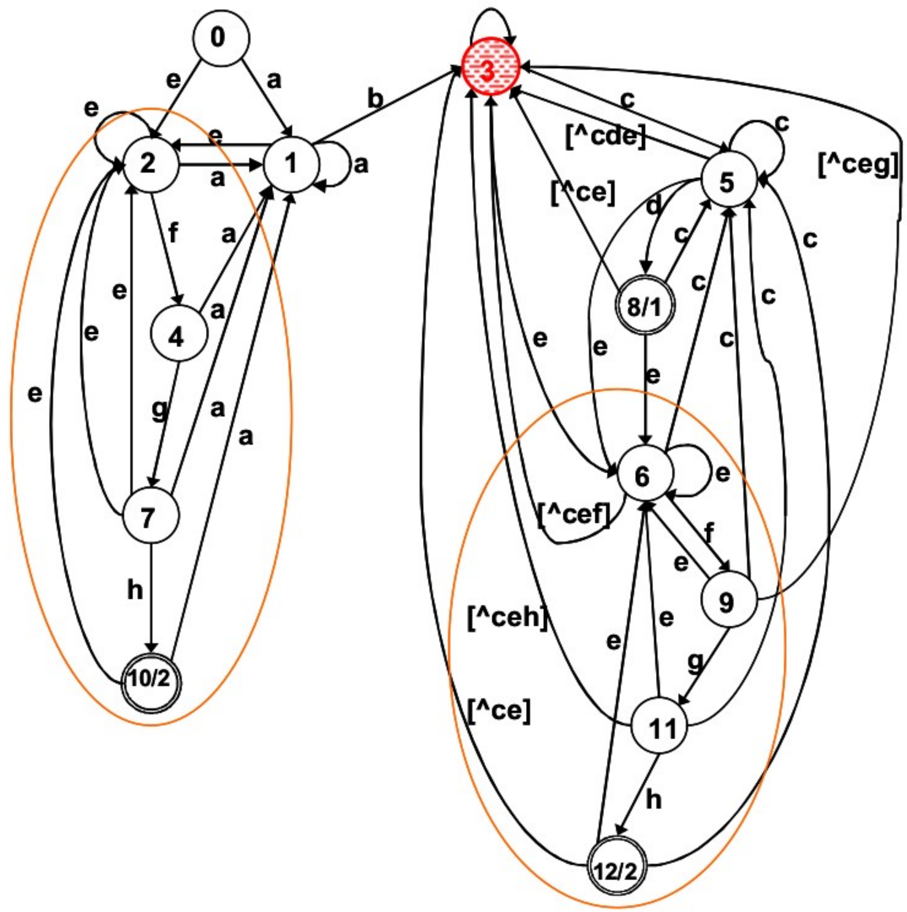
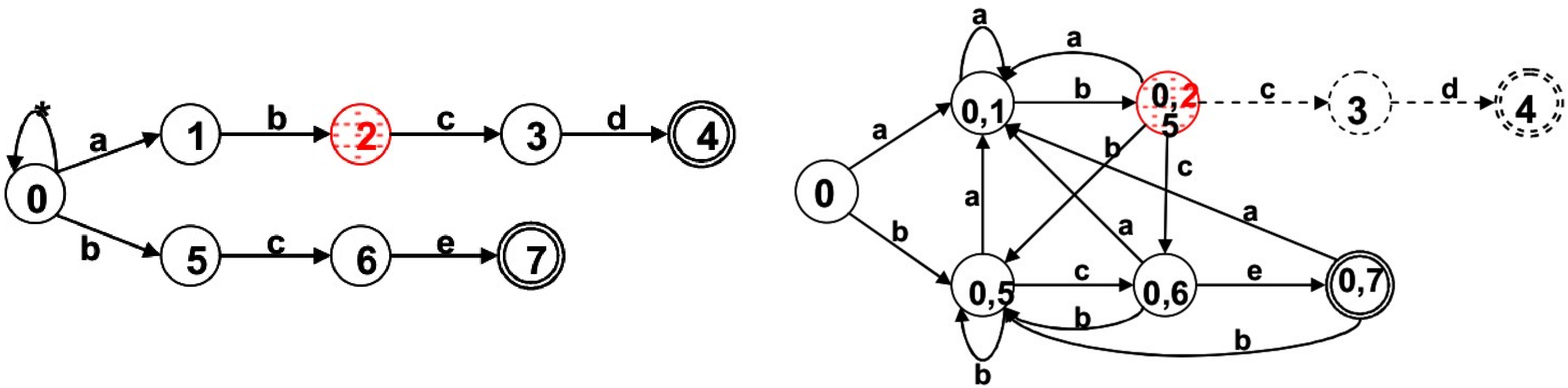


Figure 2: DFA representing (1) ab.*cd and (2) efgh. In the accepting states, the number following the "/" represents the accepted regular expression.

Approach #1:
Hybrid FA

- **Introduced by Michaela Becchi [1]**
- **Partial transformation of an NFA to a DFA**
  - Interruption of subset construction algorithm at problematic states



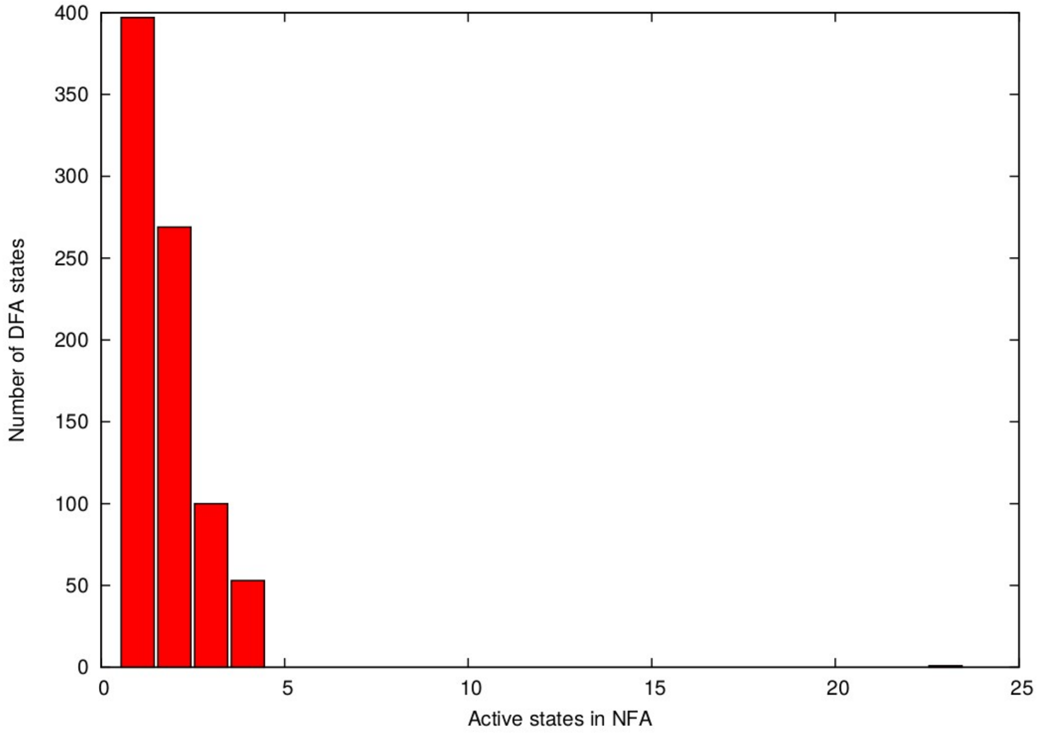
From [1]

Approach #2: System of Parallel Automaton Parts

- **Introduced by Jan Kořenek [2]**
- **Formalism to describe division of a single NFA into several parts**
- **Based on analysis of concurrency of an NFA**
- **Each part is either NFA, or DFA**
  - DFA parts deal with states of the original NFA that cannot be active concurrently
  - NFA parts deal with the other states

Analysis of NFA Concurrency: Practical cases

- **Number of concurrently active states in L7 decoder (from [3])**



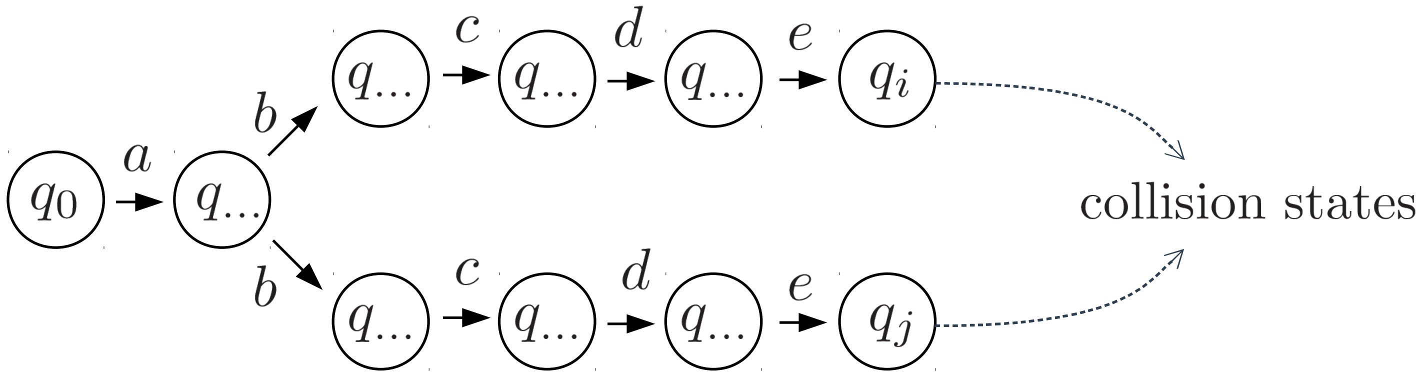
System of Parallel Automaton Parts: States without Collision

Let A be an NFA $A = (Q, \Sigma, \delta, q_0, F)$. Two states $q_i, q_j \in Q, q_i \neq g_j$ are called *states without collisoin* or *non-collision states*, if for any input string $w \in \Sigma^*$ does not exist a sequence of configurations

$$(q_0, w) \vdash^* (q_i, \varepsilon)$$

$$(q_0, w) \vdash^* (q_j, \varepsilon)$$

Example:



System of Parallel Automaton Parts: Set of States without Collision I

1. Transform NFA $A = (Q, \Sigma, \delta, q_0, F)$ to DFA $A^D = (Q^D, \Sigma, \delta^D, q_0^D, F^D)$, where $Q_D \subseteq 2^Q$.
2. For all states $q_i \in Q$ create the set $S_{q_i}^{ca}$ which contains collision states with q_i:
$$S_{q_i}^{ca} = \{q_j \in Q | q_i \neq q_j \wedge \exists q^D \in Q^D : q_i, q_j \in q^D\}$$
3. Let $Q^{nca} = Q$.

Continues on the next slide...

System of Parallel Automaton Parts: Set of States without Collision II

4. Keep removing collision states from the set Q^{nca} until the set contains only states without collisions:
   (a) select a state $q_{max} \in Q^{nca}$ with the largest set of states $S^{ca}_{q_{max}}$:

   $$\forall q_i \in Q^{nca} : |S^{ca}_{q_{max}}| \geq |S^{ca}_{q_i}|$$

   (b) remove q_{max} from Q^{nca},
   (c) for all states $q_i \in Q^{nca}$ remove q_{max} from the set $S^{ca}_{q_i}$ and
   (d) if $\exists q_i \in Q^{nca} : S^{ca}_{q_i} \neq \varnothing$ then go to (a).
5. Q^{nca} is the set of states without collision.

System of Parallel Automaton Parts: Set of States without Collision III

- **The complexity is exponential**
  - Transformation of NFA to DFA in the step 1.
- **The state to be removed in the step 4. (a) is selected based on the number of collision states (heuristic)**
  - States with the most collisions are removed first.
- **Multiple sets of states without collision can be found by recursive application of the algorithm**
  - $Q^N = Q \backslash Q^{nca}$

System of Parallel Automaton Parts: Set of States without Collision IV

- **Improved algorithm to find all pairs of simultaneously active states [4]**
- **Does not require transformation of original NFA to corresponding DFA**
- **Better complexity**

System of Parallel Automaton Parts: Set of States without Collision V

$\text{normalize}(q_1, q_2) = (q_1 < q_2)\ ?\ (q_1, q_2) : (q_2, q_1);$

$concurrent = \{(s, s)\}; workplace = \{(s, s)\};$

while $\exists (q_1, q_2) \in workplace$ **do**

  $workplace = workplace \setminus \{(q_1, q_2)\};$

  foreach $q_3 \in \delta(q_1, a)$ **do**

    foreach $q_4 \in \delta(q_2, b)$ **do**

      if $a \cap_k b \neq (\varnothing, \varnothing, \ldots, \varnothing)$ **then**

        if $((q_5, q_6) = \text{normalize}(q_3, q_4)) \notin concurrent$ **then**

          $concurrent = concurrent \cup \{(q_5, q_6)\};$

          $workplace = workplace \cup \{(q_5, q_6)\};$

return $concurrent \setminus \{(p, p) | p \in Q\};$

System of Parallel Automaton Parts: Part of the Automaton Determined by a Set of States I

Let $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA and $Q^s \subseteq Q$ is a set of states. Then the set of states Q^s determines the part of the automaton $A/_{Q^s}$, which is defined by tuple $A/_{Q^s} = (Q^s, Q_{in}, Q_{out}, \Sigma, \delta^s, q_0^s, F^s)$, where

- $Q^s \subseteq Q$ is the set of internal states.
- $Q_{in} = \{q_s | q_s \in Q^s \wedge q_s \in \delta(q, a) \wedge q \in (Q \backslash Q^s)\}$ is the set of input states.
- $Q_{out} = \{q | q \in (Q \backslash Q^s) \wedge q \in \delta(q_s, a) \wedge q_s \in Q^s\}$ is the set of output states.
- Σ is the input alphabet.

Continues on the next slide. . .

System of Parallel Automaton Parts: Part of the Automaton Determined by a Set of States II

Let $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA and $Q^s \subseteq Q$ is a set of states. Then the set of states Q^s determines the part of the automaton $A/_{Q^s}$, which is defined by tuple $A/_{Q^s} = (Q^s, Q_{in}, Q_{out}, \Sigma, \delta^s, q_0^s, F^s)$, where

- $\delta^s : Q^S \times 2^Q$ is the state-transition function restricted to the set of states Q^s. For a state $q_{src} \in Q^s$ and $q_{dst} \in Q$ and an input symbol $a \in \Sigma$ of transition $q_{dst} \in \delta^s(q_{src}, a)$ is defined only if the transition $q_{dst} \in \delta(q_{src}, a)$ is defined.
- q_0^s is the initial state of the automaton part which is defined as:

$$q_0^s = \begin{cases} q_0 & \text{for} \quad q_0 \in Q^s \\ idle & \text{for} \quad q_0 \notin Q^s \end{cases}$$

- $F^s \subseteq F$ is the set of final states restricted to Q^s: $F^s = F \cap Q^s$

System of Parallel Automaton Parts

Let $A = (Q, \Sigma, \delta, q_0, F)$ be an automaton and sets of states $Q^1, Q^2, \ldots, Q^k \subseteq Q$ determine k different parts of the automaton $A/_{Q^1}, A/_{Q^2}, \ldots, A/_{Q^k}$. *System of Parallel Automaton Parts* $A/_{[Q^1, Q^2, \ldots, Q^k]}$ is defined by set of states $Q^1, Q^2, \ldots, Q^k$, if

$$Q = \bigcup_{i=1}^{k} Q^i$$

System of Parallel Automaton Parts: Communication Models

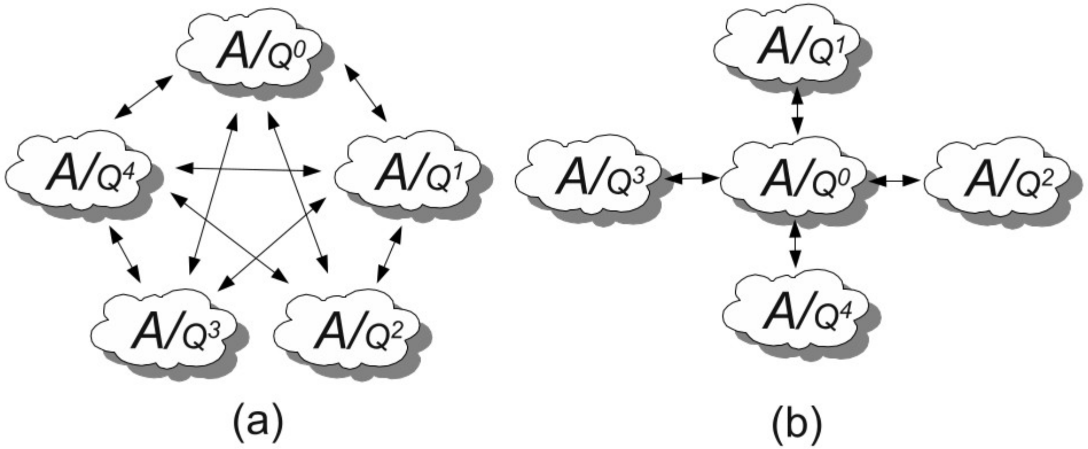
Number of bidirectional connections

- **Without central part (a)**
  - Significant communication overhead

$$\frac{k(k-1)}{2}$$

- **With central part (b)**
  - Simpler communication through central part

$$k-1$$



From [3]

System of Parallel Automaton Parts: Centralised System of Automaton Parts

Let $A/_{[Q^1,Q^2,\dots,Q^k]}$ is a System of Automaton Parts for NFA $A = (Q, \Sigma, \delta, q_0, F)$. The System is called *centralised* if for any set of states $Q^j, j \in< 1; k >$ it holds:

1. $\forall i \in< 1; k >, i \neq j : (Q^i \cap Q^j) = \emptyset$
2. $\forall i \in< 1; k >, i \neq j : (Q^i_{in} \subseteq Q^j_{out})$
3. $\forall i \in< 1; k >, i \neq j : (Q^i_{out} \subseteq Q^j_{in})$

Then $A/_{Q^j}$ is called a *central part* or a *central item* of the centralised system $A/_{[Q^1,Q^2,\dots,Q^k]}$.

System of Parallel Automaton Parts: Tranformation to Centralised System

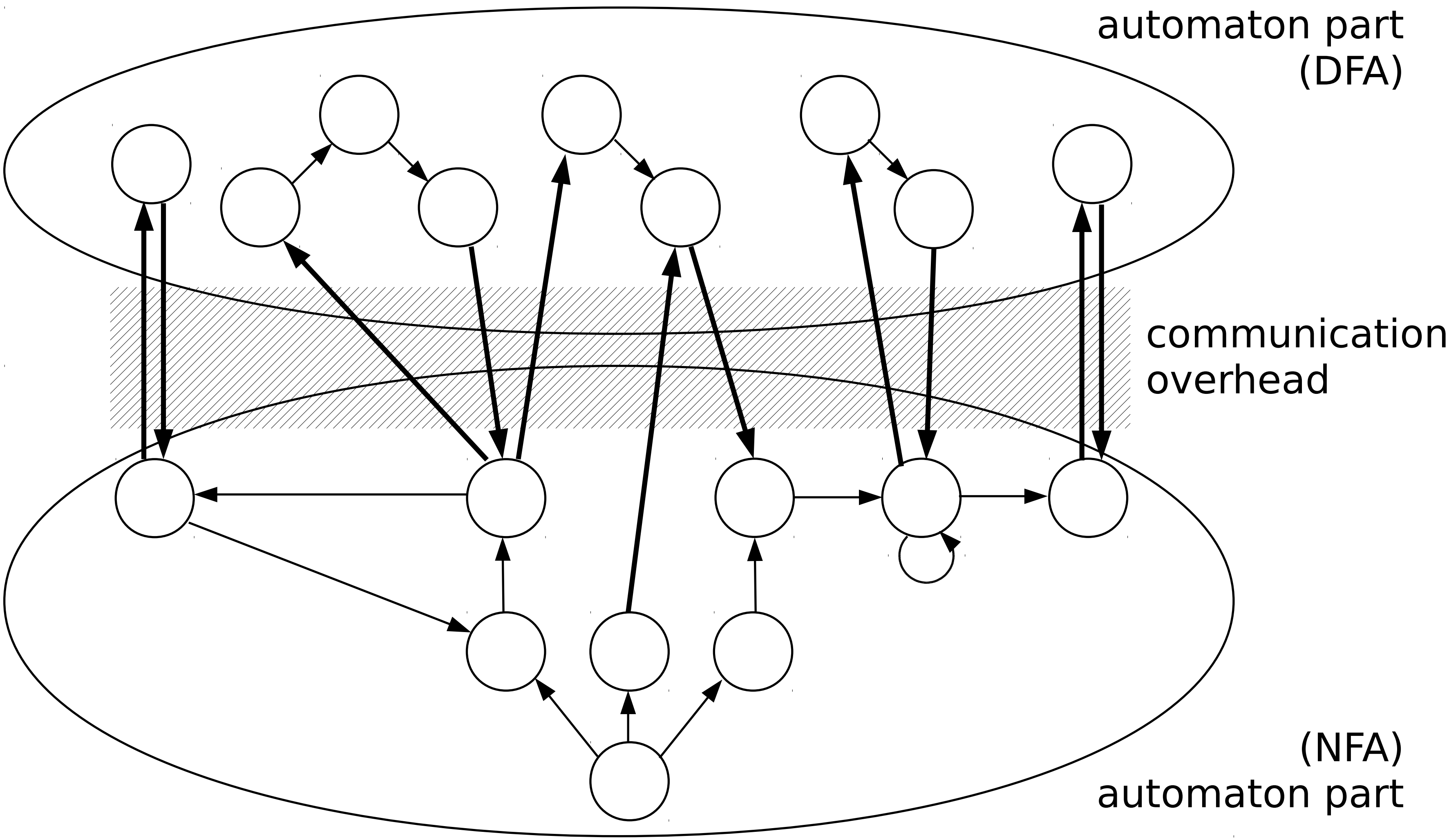
1. Let $\forall i \in \langle 1; k \rangle, i \neq r : Q^{c_i} = Q_i^{nca} \setminus Q_r^{nca}$
2. Let $Q^{c_N} = Q^N$
3. For all $i \in \langle 1; k \rangle, i \neq r$ do:
   (a) $Q^{c_N} = Q^{c_N} \cup Q_{out}^{c_i}$
   (b) $\forall j \in \langle 1; k \rangle, j \neq r : Q^{c_j} \setminus Q_{out}^{c_i}$
4. The system $A/_{[Q^{c_N}, Q^{c_1}, Q^{c_2}, \ldots, Q^{c_k}]}$ is centralised and $A/_{Q^{c_N}}$ is the central part.

- **The algorithm moves all output states of all parts to the central part.**

System of Parallel Automaton Parts: Issues I

- **The algorithm to find a set of states without collision is applied recursively**
- **The issue is that the set of states obtained with the first application of the algorithm:**
  - contains much more states than the sets of states obtained by another applications of the algorithm,
  - has many isolated groups of states, which causes significant communication overhead.

System of Parallel Automaton Parts: Issues II



References

[1] Michela Becchi and Patrick Crowley. A Hybrid Finite Automaton for Practical Deep Packet Inspection. In *Proceedings of the International Conference on emerging Networking Experiments and Technologies (CoNEXT)*, New York, NY, December 2007. ACM.

[2] Jan Kořenek. Rychlé vyhledávání regulárních výrazů s využitím technologie FPGA, disertační práce, Brno, FIT VUT v Brně, 2010

[3] Jan Kořenek. Fast Regular Expression Matching Using FPGA. *Information Sciences and Technologies Bulletin of the ACM Slovakia.* Bratislava: Vydavateľstvo STU, 2010, vol. 2, no. 2, pp. 103-111. ISSN 1338-1237.

[4] KOŠAŘ Vlastimil and KOŘENEK Jan. Multi-Stride NFA-Split Architecture for Regular Expression Matching Using FPGA. In: *Proceedings of the 9th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science.* Brno: NOVPRESS s.r.o., 2014, s. 77-88. ISBN 978-80-214-5022-6.