A Concurrent Specification of an Incremental DFA Minimisation Algorithm

Tinus Strauss Derrick G. Kourie Bruce W. Watson
FASTAR, University of Pretoria
http://www.fastar.org

September 3, 2008

Outline

Introduction

Research Drivers

- Increase in FA size
- Multiple CPUs on single die

Research Approach

- Select classical sequential FA algorithm
- Provide CSP fine-grained parallelized equivalent
- Implement and test different granularities

Where are we?

- Brzozowski's FA construction from Regex
  ⇒ CSP done; Erlang implementation underway
- Watson / Daciuk FA Minimization
  ⇒ CSP done; New CSP operator proposed

Introduction

Research Drivers

- Increase in FA size
- Multiple CPUs on single die

Research Approach

- Select classical sequential FA algorithm
- Provide CSP fine-grained parallelized equivalent
- Implement and test different granularities

Where are we?

- Brzozowski's FA construction from Regex
  $\Rightarrow$ CSP done; Erlang implementation underway
- Watson / Daciuk FA Minimization
  $\Rightarrow$ CSP done; New CSP operator proposed

Introduction

Research Drivers

- Increase in FA size
- Multiple CPUs on single die

Research Approach

- Select classical sequential FA algorithm
- Provide CSP fine-grained parallelized equivalent
- Implement and test different granularities

Where are we?

- Brzozowski's FA construction from Regex
  $\Rightarrow$ CSP done; Erlang implementation underway
- Watson / Daciuk FA Minimization
  $\Rightarrow$ CSP done; New CSP operator proposed

Preliminaries

Notation

- *DFA*: $(Q, \Sigma, \delta, q_0, F)$
- Out-transitions of q: Σ_q
- Left language of q : $\overrightarrow{\mathcal{L}}(q) = \{ w \mid \delta^*(q, w) \in F \}$
- $Equiv(p, q)$:
  - Semantically: $\overrightarrow{\mathcal{L}}(p) = \overrightarrow{\mathcal{L}}(q)$
  - Recursively: $(p \in F \equiv q \in F) \wedge (\Sigma_p = \Sigma_q) \wedge \langle \forall\ a \in \Sigma_{pq} : Equiv(\delta(p, a), \delta(q, a)) \rangle$
- DFA minimal iff $\langle \forall\ p, q \in Q : p \neq q : \neg Equiv(p, q) \rangle$

Preliminaries

Notation

- *DFA*: $(Q, \Sigma, \delta, q_0, F)$
- Out-transitions of q: Σ_q
- Left language of q : $\overrightarrow{\mathcal{L}}(q) = \{ w \mid \delta^*(q, w) \in F \}$
- $Equiv(p, q)$:
  - Semantically: $\overrightarrow{\mathcal{L}}(p) = \overrightarrow{\mathcal{L}}(q)$
  - Recursively: $(p \in F \equiv q \in F) \wedge (\Sigma_p = \Sigma_q) \wedge \langle \forall\ a \in \Sigma_{pq} : Equiv(\delta(p, a), \delta(q, a)) \rangle$
- DFA minimal iff $\langle \forall\ p, q \in Q : p \neq q : \neg Equiv(p, q) \rangle$

Preliminaries

Notation

- *DFA*: $(Q, \Sigma, \delta, q_0, F)$
- Out-transitions of q: Σ_q
- Left language of q : $\overrightarrow{\mathcal{L}}(q) = \{ w \mid \delta^*(q, w) \in F \}$
- $Equiv(p, q)$:
  - Semantically: $\overrightarrow{\mathcal{L}}(p) = \overrightarrow{\mathcal{L}}(q)$
  - Recursively: $(p \in F \equiv q \in F) \wedge (\Sigma_p = \Sigma_q) \wedge$ $\langle \forall\ a \in \Sigma_{pq} : Equiv(\delta(p, a), \delta(q, a)) \rangle$
- DFA minimal iff $\langle \forall\ p, q \in Q : p \neq q : \neg Equiv(p, q) \rangle$

Preliminaries

Notation

- *DFA*: $(Q, \Sigma, \delta, q_0, F)$
- Out-transitions of q: Σ_q
- Left language of q : $\overrightarrow{\mathcal{L}}(q) = \{ w \mid \delta^*(q, w) \in F \}$
- $Equiv(p, q)$:
  - Semantically: $\overrightarrow{\mathcal{L}}(p) = \overrightarrow{\mathcal{L}}(q)$
  - Recursively: $(p \in F \equiv q \in F) \wedge (\Sigma_p = \Sigma_q) \wedge$ $\langle \forall\ a \in \Sigma_{pq} : Equiv(\delta(p, a), \delta(q, a)) \rangle$
- DFA minimal iff $\langle \forall\ p, q \in Q : p \neq q : \neg Equiv(p, q) \rangle$

Preliminaries

Notation

- *DFA*: $(Q, \Sigma, \delta, q_0, F)$
- Out-transitions of q: Σ_q
- Left language of q : $\overrightarrow{\mathcal{L}}(q) = \{ w \mid \delta^*(q, w) \in F \}$
- $Equiv(p, q)$:
  - Semantically: $\overrightarrow{\mathcal{L}}(p) = \overrightarrow{\mathcal{L}}(q)$
  - Recursively: $(p \in F \equiv q \in F) \wedge (\Sigma_p = \Sigma_q) \wedge$ $\langle \forall\ a \in \Sigma_{pq} : Equiv(\delta(p, a), \delta(q, a)) \rangle$
- DFA minimal iff $\langle \forall\ p, q \in Q : p \neq q : \neg Equiv(p, q) \rangle$

Preliminaries

Notation

- *DFA*: $(Q, \Sigma, \delta, q_0, F)$
- Out-transitions of q: Σ_q
- Left language of q : $\overrightarrow{\mathcal{L}}(q) = \{ w \mid \delta^*(q, w) \in F \}$
- $Equiv(p, q)$:
  - Semantically: $\overrightarrow{\mathcal{L}}(p) = \overrightarrow{\mathcal{L}}(q)$
  - Recursively: $(p \in F \equiv q \in F) \wedge (\Sigma_p = \Sigma_q) \wedge \langle \forall\ a \in \Sigma_{pq} : Equiv(\delta(p, a), \delta(q, a)) \rangle$
- DFA minimal iff $\langle \forall\ p, q \in Q : p \neq q : \neg Equiv(p, q) \rangle$

Preliminaries

Notation

- *DFA*: $(Q, \Sigma, \delta, q_0, F)$
- Out-transitions of q: Σ_q
- Left language of q : $\overrightarrow{\mathcal{L}}(q) = \{\, w \mid \delta^*(q, w) \in F \,\}$
- $Equiv(p, q)$:
  - Semantically: $\overrightarrow{\mathcal{L}}(p) = \overrightarrow{\mathcal{L}}(q)$
  - Recursively: $(p \in F \equiv q \in F) \wedge (\Sigma_p = \Sigma_q) \wedge$ $\langle \forall\ a \in \Sigma_{pq} : Equiv(\delta(p, a), \delta(q, a)) \rangle$
- DFA minimal iff $\langle \forall\ p, q \in Q : p \neq q : \neg Equiv(p, q) \rangle$

Preliminaries

Notation

- *DFA*: $(Q, \Sigma, \delta, q_0, F)$
- Out-transitions of q: Σ_q
- Left language of q : $\overrightarrow{\mathcal{L}}(q) = \{\, w \mid \delta^*(q, w) \in F \,\}$
- $Equiv(p, q)$:
  - Semantically: $\overrightarrow{\mathcal{L}}(p) = \overrightarrow{\mathcal{L}}(q)$
  - Recursively: $(p \in F \equiv q \in F) \wedge (\Sigma_p = \Sigma_q) \wedge$ $\langle \forall\ a \in \Sigma_{pq} : Equiv(\delta(p, a), \delta(q, a)) \rangle$
- DFA minimal iff $\langle \forall\ p, q \in Q : p \neq q : \neg Equiv(p, q) \rangle$

The Sequential Algorithm

Algorithm 3.1 (Computing *Equiv*):

$S, G, H := \emptyset, ((Q \setminus F) \times F) \cup (F \times (Q \setminus F)), \{ (q, q) \mid q \in Q \};$
$\{$ invariant: $G \subseteq \neg Equiv \wedge H \subseteq Equiv$ $\}$
do $(G \cup H) \neq Q \times Q \rightarrow$
    let $p, q : (p, q) \in ((Q \times Q) \setminus (G \cup H));$
    if $equiv(p, q, (|Q| - 2)\,\mathbf{max}\,0) \rightarrow$
        $H := H \cup \{(p, q), (q, p)\};$
        $H := H^+$
    $[\!]$ $\neg equiv(p, q, (|Q| - 2)\,\mathbf{max}\,0) \rightarrow$
        $G := G \cup \{(p, q), (q, p)\}$
    fi
od; $\{$ $H = Equiv$ $\}$
merge states according to H
$\{$ $(Q, \Sigma, \delta, q_0, F)$ is minimal $\}$

The Sequential Algorithm

Algorithm 3.2 (Pointwise computation of $Equiv(p, q)$):

$$
\begin{array}{l}
\textbf{func}\ \ equiv(p,q,k) \rightarrow \\
\quad\quad \textbf{if}\ \ k = 0 \rightarrow eq := (p \in F \equiv q \in F) \\
\quad\quad [\!]\ \ k \neq 0 \wedge \{p,q\} \in S \rightarrow eq := true \\
\quad\quad [\!]\ \ k \neq 0 \wedge \{p,q\} \notin S \rightarrow \\
\quad\quad\quad\quad eq := (p \in F \equiv q \in F) \wedge (\Sigma_p = \Sigma_q); \\
\quad\quad\quad\quad S := S \cup \{\{p,q\}\}; \\
\quad\quad\quad\quad \textbf{for}\ \ (a \in \Sigma_p \cap \Sigma_q) \rightarrow \\
\quad\quad\quad\quad\quad\quad eq := eq \wedge equiv(\delta(p,a), \delta(q,a), k-1) \\
\quad\quad\quad\quad \textbf{rof}; \\
\quad\quad\quad\quad S := S \setminus \{\{p,q\}\} \\
\quad\quad \textbf{fi}; \\
\quad\quad \textbf{return}\ \ eq \\
\textbf{cnuf}
\end{array}
$$

CSP Background

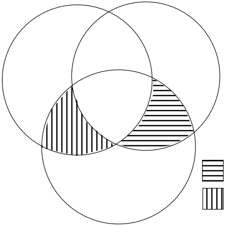
Selected CSP Notation

$a \rightarrow P$	event a then process P
$a \rightarrow P \| b \rightarrow Q$	a then P choice b then Q
$x?A \rightarrow P(x)$	choice of x from set A then $P(x)$
$P \underset{X}{\|\|} Q$	P in parallel with Q; Sync on X
$b!e$	on channel b output event e
$b?x$	from channel b input to variable x
$P; Q$	process P followed by process Q
$P \|\|\| Q$	process P interleave process Q
$P \triangle Q$	process P interrupted by process Q

CSP: $R_1 \underset{X}{\parallel} R_2$

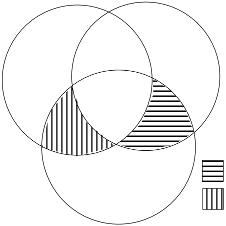
- $P \ = \ y \rightarrow P'$
- $R_1 \ = ?x : A_1 \rightarrow R_1'$
- $R_2 \ = ?x : A_2 \rightarrow R_2'$

- $P \underset{X}{\parallel} (R_1 \underset{X}{\parallel} R_2) = (y \rightarrow (P' \underset{X}{\parallel} (R_1' \underset{X}{\parallel} R_2')))$ iff $y \in X \cap (A_1 \cap A_2)$

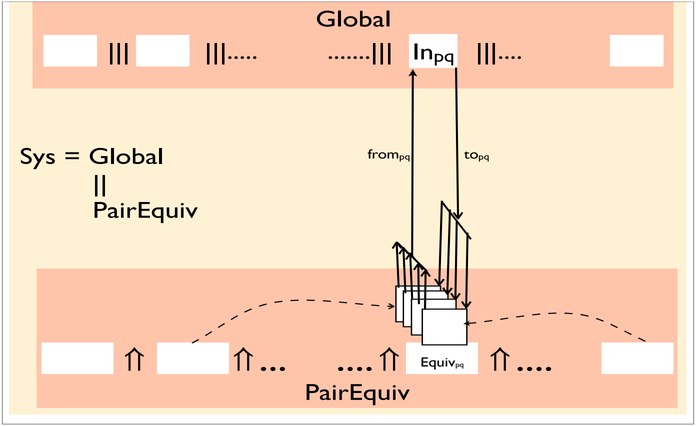


CSP: $R_1 \underset{X}{\|} R_2$ vs $R_1 \underset{X}{\Uparrow} R_2$

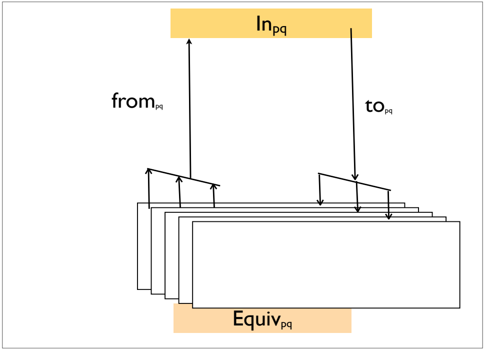
- $P \ = \ y \rightarrow P' \qquad R_1 \ = ?x : A_1 \rightarrow R_1' \qquad R_2 \ = ?x : A_2 \rightarrow R_2'$
- $P \underset{X}{\|} (R_1 \underset{X}{\Uparrow} R_2) = y \rightarrow (P' \underset{X}{\|} (R_1' \underset{X}{\Uparrow} R_2'))$ iff $y \in X \cap (A_1 \cup A_2)$
- $P \underset{X}{\|} (R_1 \underset{X}{\Uparrow} R_2) = y \rightarrow (P' \underset{X}{\|} (R_1' \underset{X}{\Uparrow} R_2))$ iff $y \in (X \cap A_1) \backslash A_2$
- $P \underset{X}{\|} (R_1 \underset{X}{\Uparrow} R_2) = y \rightarrow (P' \underset{X}{\|} (R_1 \underset{X}{\Uparrow} R_2'))$ iff $y \in (X \cap A_2) \backslash A_1$



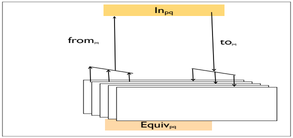
$$
\begin{aligned}
Global &= |||_{(p,q)\in P}\ In_{pq} \\
PairEquiv &= \underset{\alpha}{\Uparrow}_{(p,q)\in P}\ Equiv_{pq}(\emptyset, (|Q|-2)\ \mathbf{max}\ 0)
\end{aligned}
$$



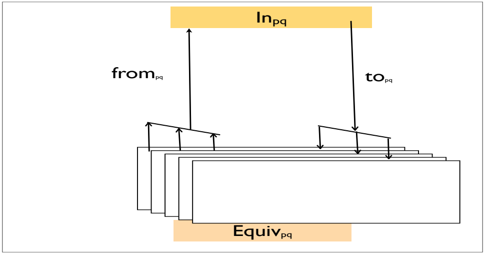
$$
\begin{aligned}
In_{pq} &= from_{pq}?e \rightarrow Announce_{pq}(e) \\
Announce_{pq}(e) &= (to_{pq}!e \rightarrow Announce_{pq}(e) \\
&\quad \mid from_{pq}?e \rightarrow Announce_{pq}(e))
\end{aligned}
$$



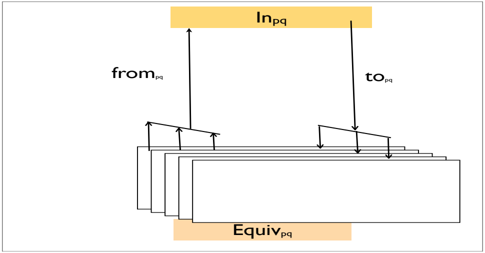
$$
\begin{aligned}
& Equiv_{pq}(S,k) = \\
& \quad \text{if } (k = 0) \text{ then } from_{pq}!(p \in F \equiv q \in F) \rightarrow SKIP \\
& \quad \text{else if } (p = q) \text{ then } from_{pq}!true \rightarrow SKIP \\
& \quad \text{else } \boxed{\text{if } ((p \in F \equiv q \in F) \wedge (\Sigma_p = \Sigma_q)) \text{ then }} (\\
& \quad EqSet := \emptyset \\
& \quad ;\ FanOut_{pq}(S \cup \{(p,q)\}, k) \\
& \quad ;\ (eq := \textstyle\bigwedge_{e \in EqSet} e) \\
& \quad ;\ (from_{pq}!eq \rightarrow SKIP)) \\
& \quad \boxed{\text{else } from_{pq}!false \rightarrow SKIP}
\end{aligned}
$$



$$
\begin{aligned}
&FanOut_{pq}(S,k) = \\
&\qquad |||_{a \in \Sigma_{pq}} (\\
&\qquad\qquad \text{if } (\{\delta(p,a), \delta(q,a)\} \notin S) \text{ then } REquiv_{pq}(S,k,a) \\
&\qquad\qquad \text{else } (EqSet := EqSet \cup \{true\}) \\
&\qquad\qquad)
\end{aligned}
$$



$$
\begin{aligned}
REquiv_{pq}(S, k, a) = \\
& (u, v := \delta(p, a), \delta(q, a) \\
& ; Equiv_{uv}(S, k-1) \\
& \boxed{; (to_{uv}?eq_a \rightarrow (EqSet := EqSet \cup \{eq_a\})}) \\
& \triangle \\
& (to_{uv}?eq_a \rightarrow (EqSet := EqSet \cup \{eq_a\}))
\end{aligned}
$$



Conclusion

- Many optimisation possibilities
- Now for implemention . . .