Minimizing NFA's and Regular Expressions

Gregor Gramlich and Georg Schnitger
Institute of Computer Science
Johann Wolfgang Goethe-Universität Frankfurt

February 2005

Minimization Problems

Problems of **exactly** determining the minimum size of an equivalent NFA or regular expression for a given NFA, regular expression or DFA.

$$\text{Regular Expression} \rightarrow \text{Regular Expression}$$
$$\text{NFA} \rightarrow \text{NFA}$$

are PSPACE-complete. (Meyer and Stockmeyer, 1972)

Also known to be PSPACE-complete

$$\text{DFA} \rightarrow \text{NFA. (Jiang and Ravikumar, 1993)}$$

Approximation Problems

Are efficient and tight **approximations** of small NFAs and Regular Expressions possible . . .

. . . when given an NFA or regular expression?

. . . when given a DFA?

Results

$$\begin{tikzcd} \text{Regular Expression} \arrow[r] \arrow[dr] & \text{Regular Expression} \\ \text{NFA} \arrow[r] \arrow[ur] & \text{NFA} \end{tikzcd}$$

can only have an efficient approximation with factor $\mu = o(n)$ for input size n, if $P =$PSPACE.

Every efficient approximation algorithm for the problems

$$\begin{array}{lcl} \text{DFA} & \longrightarrow & \text{Regular Expression} \\ \text{DFA} & \longrightarrow & \text{NFA} \end{array}$$

must have an approximation factor of at least $\frac{n}{\mathrm{poly}(\log n)}$ for given DFAs with n states, if strong pseudo-random functions exist in NC^1.
(The size of an NFA is the number of transitions.)

Sublinear Approximation is PSPACE-hard

The transformation used to prove PSPACE-hardness of the non-universality problem $L(R) \stackrel{?}{\neq} \Sigma^*$ can be made gap introducing.

Theorem 1. For given NFA or regular expression with n states, transitions or symbols respectively, it is impossible to efficiently approximate the size of a minimal equivalent NFA or regular expression within an approximation factor of $o(n)$, if $P \neq$ PSPACE.

This is true for regular expressions and NFAs over an alphabet with at least two symbols.

The unary case ($|\Sigma| = 1$) has to be treated differently.

Unary NFA and Regular Expression Minimization

Given a unary NFA or a unary regular expression of size n, it is impossible to efficiently **approximate the minimal size** of an equivalent NFA or regular expression within a factor of $\frac{\sqrt{n}}{\ln n}$, if $P \neq NP$.
(Gramlich, 2003)

If we require the **construction** of an approximately minimal regular expression or NFA, we can exclude even higher approximation factors.

Theorem 2. Given an arbitrary $\delta > 0$ and a unary NFA or a unary regular expression of size n, it is impossible to efficiently **construct** an equivalent NFA or regular expression within approximation factor $n^{1-\delta}$, if $P \neq NP$.

Minimal NFAs and Regular Expressions for Given DFAs

The problem

$$\textbf{DFA} \quad \longrightarrow \quad \text{NFA}$$

is PSPACE-complete, but the transformation in the proof (Jiang and Ravikumar, 1993) is not gap introducing.

Strong Pseudo-Random Functions

There is an NC^1 function ensemble f_m, such that for any randomized algorithm A

$$|\text{prob}[A(f_m) = 1] - \text{prob}[A(r_m) = 1]| < \frac{1}{3},$$

provided A runs in time $2^{O(m)} = \text{poly}(2^m)$ and factorization is sufficiently hard. (f_m pseudo-random, r_m truly random m-bit function)

A has access to the full truth table of f_m, resp. r_m.

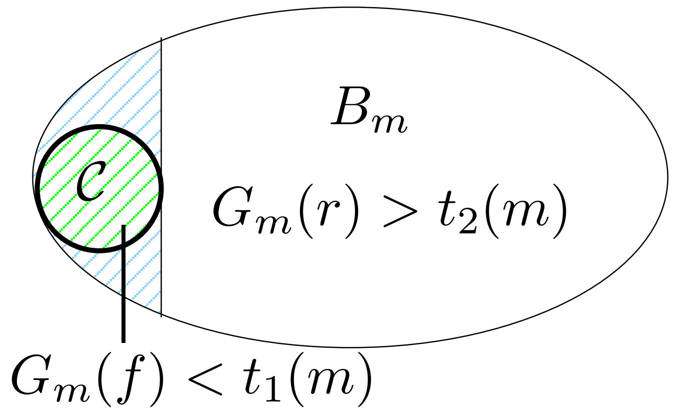
We call such a function ensemble a strong pseudo-random ensemble.

If strong pseudo-random functions in the sense of Razborov & Rudich exist, then strong pseudo-random functions in our sense exist.

Inapproximability and Pseudo-Random Functions

- Functional $G_m : B_m \to \mathbb{N}$, measures the complexity of a function.
- Idea: $G_m(f) =$ size of a minimal regular expression for $L(f) = \{x | f(x) = 1\}$.
- $G = (G_m)_m$ separates a function class $\mathcal{C}$ from random functions with thresholds $t_1(\cdot)$ and $t_2(\cdot)$, if

$$\begin{aligned} \forall f \in \mathcal{C} \cap B_m : \quad & G_m(f) < t_1(m), \text{ and} \\ |\{r \in B_m \mid G_m(r) \le t_2(m)\}| \; = \; & o(|B_m|). \end{aligned}$$



If $\mathcal{C}$ contains a strong pseudo-random ensemble, then no approximation algorithm for G with running time $2^{O(m)}$ can have an approximation factor smaller than $\frac{t_2(m)}{t_1(m)}$.

Formulae of Logarithmic Depth

There is a strong pseudo-random ensemble $\mathcal{C}_1$ in NC^1 with formula-depth $c \cdot \log m$ and formula-length m^c for input size m and some constant c.

Formula: Complete binary tree, leaves are positive or negative literals. (Negations are pushed into the leaves.)

The length ℓ of a formula is the number of leaves. The depth d of a formula is the depth of the tree.

$$\ell = 2^d.$$

Regular Expressions for Short Formulae

- Goal: Express a formula $\mathbf{f}$ of small depth by a short regular expression.
- Problem: Regular expressions are too weak.
- Solution: Repeat inputs. Instead of expressing $L(f) = \{x|f(x) = 1\}$, express $L_k(f) := \{x^k|f(x) = 1\}$.

For a formula $\mathbf{f}$ of depth $c \cdot \log m$ for $f \in B_m$, there is a regular expression $R_\mathbf{f}$ of length $O(m^{2c+1})$, such that if we promise to repeat inputs, then $L(R_\mathbf{f}) = L_{m^c}(f_m)$:

$$L(R_\mathbf{f}) \cap \{x^*|x \in \{0,1\}^m\} \quad = \quad \{x^{m^c}|f_m(x) = 1\} \quad = \quad L_{m^c}(f_m).$$

Assigning Regular Expression $R_{\mathbf{f}}$ to $\mathbf{f}$

- If $\mathbf{f} = x_i$, then $R_{\mathbf{f}} := (0+1)^{i-1}\, 1\, (0+1)^{m-i}$.
- If $\mathbf{f} = \overline{x_i}$, then $R_{\mathbf{f}} := (0+1)^{i-1}\, 0\, (0+1)^{m-i}$.
- If $\mathbf{f} = \mathbf{f}_1 \wedge \mathbf{f}_2$, then $R_{\mathbf{f}} := R_{\mathbf{f}_1} \circ R_{\mathbf{f}_2}$.
- If $\mathbf{f} = \mathbf{f}_1 \vee \mathbf{f}_2$, then $R_{\mathbf{f}} := R_{\mathbf{f}_1} \circ (0+1)^{m\cdot\ell(\mathbf{f}_2)} + (0+1)^{m\cdot\ell(\mathbf{f}_1)} \circ R_{\mathbf{f}_2}$.

$L_{m^c}(f_m) = L(R_{\mathbf{f}}) \cap \{x^*|x \in \{0,1\}^m\}$ holds. But how to check, whether the promise of repeated inputs is kept?

The complement $\overline{L_{m^c}(f_m)} = \overline{L(R_{\mathbf{f}})} \cup \overline{\{x^*|x \in \{0,1\}^m\}}$ is easy to check and has a regular expression of length $O(m^{2c+1})$.

Approximation complexity for DFA $\to$ Regular Expression I

- Let $G_m(f_m)$ be the size of a smallest regular expression for $\overline{L_{m^c}(f_m)}$.

- Thus
$$G_m(f_m) \leq t_1(m) = O(m^{2c+1})$$
holds for functions with formula depth $c \cdot \log m$.

- There are only $o(|B_m|)$ different regular expressions of length at most $2^m/40$. So
$$G_m(r_m) > t_2(m) = 2^m/40$$
holds for the vast majority of functions $r_m \in B_m$.

- Every efficient approximation algorithm for G_m must have an approximation factor of at least $\frac{t_2(m)}{t_1(m)} = \frac{2^m}{\mathrm{poly}(m)}$.
The input for G_m is a truth table, but where is the DFA?

Approximation complexity for DFA $\rightarrow$ Regular Expression II

Approximation of $G_m(f_m)$ better than $\frac{2^m}{\mathrm{poly}(m)}$ is hard, so approximation of DFA $\rightarrow$ Regular Expression is hard!

$$
\begin{tikzcd}
\boxed{f_m} \arrow[r, "\text{Transformation}"] \arrow[rr, bend right, "G_m \text{ Approx}"'] & \boxed{\text{DFA } D_{f_m}} \arrow[r, "\text{DFA}\rightarrow\text{R.E. Approx}"] & \boxed{s \in \mathbb{N}}
\end{tikzcd}
$$

DFA D_{f_m} accepts $\overline{L_{m^c}(f_m)} = \overline{\{x^{m^c} | f_m(x) = 1\}}$ with "only"

$$n = m^c \cdot 2^m = 2^{O(m)}$$

states.

Approximation complexity for DFA $\rightarrow$ Regular Expression III

Factor $\mu < \frac{2^m}{\mathrm{poly}(m)}$ is excluded, where m is the number of input bits of f_m.

Translate from m to DFA size $n = 2^{O(m)}$.

Theorem 3. Any efficient approximation algorithm for the DFA $\rightarrow$ Regular Expression (NFA, states) problem must have an approximation factor $\mu \geq \frac{n}{\mathrm{poly}(\log n)}$ $(\frac{\sqrt{n}}{\mathrm{poly}(\log n)})$ for a given DFA of size n.

Conclusions

- The problems

$$\begin{array}{rcl} \text{Regular Expression} & \rightarrow & \text{Regular Expression} \\ \text{NFA} & \rightarrow & \text{NFA} \end{array}$$

  can only have an efficient approximation with factor $\mu = o(n)$ for input size n, if $P =$PSPACE.

- In the unary case, for any $\delta > 0$, **constructive** approximation algorithms, which output a small equivalent regular expression or NFA, can only have an efficient approximation with factor $\mu < n^{1-\delta}$, if $P = NP$.

Conclusions Continued

- Every efficient approximation algorithm for

$$\begin{aligned} \text{DFA} &\longrightarrow \text{Regular Expression} \\ \text{DFA} &\longrightarrow \text{NFA} \end{aligned}$$

  must have an approximation factor $\mu \geq \frac{n}{\text{poly}(\log n)}$, resp. ($\mu \geq \frac{\sqrt{n}}{\text{poly}(\log n)}$) for given DFAs with n states, if strong pseudo-random functions exist in NC^1.

- Every efficient approximation algorithm for the minimum consistent DFA problem must have an approximation factor of at least $\frac{n}{\text{poly}(\log n)}$ for n given examples, if strong pseudo-random functions exist in DSPACE($\log n$).

Open Problems

- Are cryptographic assumptions required or are weaker assumptions like $P \neq NP$ sufficient to show inapproximability for DFA $\longrightarrow$ NFA / R.E.?

- How hard is Truth Table $\longrightarrow$ NFA approximation (minimal NFA for $\{x|f(x) = 1\}$)?

- What is the approximation complexity of the Unary DFA $\longrightarrow$ NFA problem?

  No NP-hardness results known, but exact minimization is not in P, unless $NP \subseteq \mathrm{DTIME}(n^{O(\log n)})$. (Jiang, McDowell and Ravikumar, 1991)

  The cyclic case can be approximated within $1 + \ln n$. (Gramlich, 2003)