第 1 章

おわりに

1.1 まとめ

正規表現を非脆弱化するプログラム REMEDY と言語等価性判定器 RegEq を用いて，ReDoS 脆弱な正規表現を意味的に等価でかつ非脆弱なものに修正する手法を提案した．また，そのアルゴリズムを実装し，プログラムを走らせることで ReDoS 脆弱な正規表現を意味的な等しさを保ったまま非脆弱化することができた．

1.2 今後の課題

今回のアルゴリズムでは REMEDY への初回の入力を修正したい正規表現としていた．そうではなく次のような手法を試したい．まずはじめに RegEq にて ε と修正したい正規表現の比較を行い，そこで得られた反例と ε を REMEDY への入力とする．得られた正規表現を修正したい正規表現と RegEq にて比較し，反例を得る．この工程を繰り返し行う．ε を基準に修正したい正規表現と等価な正規表現を生成していく流れとなる．ε から修正したい正規表現へと近づけていくのでよりサイズが小さく非脆弱な正規表現が得られる可能性がある．

また，RegEq の実装上，比較する 2 つの正規表現をどちらも DFA に変換している．しかし，本アルゴリズムでは修正元の正規表現は 1 回 DFA 化するだけで十分である．ループが繰り返されるたびに DFA 化するのは非効率的なので今後改良したい．

さらに，そもそも等価性判定のアルゴリズムとして今回採用したものが適しているのかも疑問に残る．理由は 2 つある．1 つ目は今回使用したアルゴリズムより効率よく等価性判定を行うことのできるものが存在するかもしれないからである．2 つ目は等価性判定器

が等価でないと決定する際に得られる反例が良いものでない可能性があるからである．しかし，現段階ではどのような例を REMEDY に渡すとより小さくより速く非脆弱な正規表現が得られるのかよく分かっていないので今後の課題としたい．

最後に，**??**節でも述べたとおり，character set を選択 | でつなげた形に変換していた．この方法は計算量的に恐らく好ましくない．RegEq を character set を直接扱えるように変更する，ないしは character set を直接扱えるような言語等価性判定器を 1 から自作する必要があるだろう．