第 1 章

提案手法

本章では ReDoS 脆弱な正規表現の修正方法を述べる．

1.1 既存の修正方法の問題点

まず，REMEDY は，文字列例を正しく分類し，RWS1U を満たし，入力の正規表現に構文的に近い正規表現を出力として生成するのだった．ここで出力の正規表現は少なくとも入力として与えられた文字列例だけは正しく分類する能力はあるが，その他の文字列については入力の正規表現と同じように分類するかは分からないことに注意されたい．つまり，REMEDY の入力の正規表現と出力の正規表現は意味的に等価でない可能性がある．

正規表現を意味的に等しくかつ非脆弱なものへと修正する方法は既に存在する．[1] では，正規表現と等価な NFA を構成，そして DFA に変換し，その DFA を等価で strongly unambiguous な正規表現に戻すという手法を用いている．しかし，この手法は [1] でも言及されているように，修正後の正規表現のサイズが入力の正規表現のサイズに対して二重指数関数的になる場合もある．正規表現のサイズは文字列のパターンマッチングの処理時間に大きく関係するので，この手法は計算量の観点から望ましくない．

1.2 ReDoS 脆弱な正規表現の新たな修正手法

正規表現のサイズをできるだけ小さく保ったまま，元の正規表現と意味的に等価でかつ非脆弱であるように修正したい．本節ではその修正手法を紹介する．

1.2.1 使用するプログラム

新たな提案手法を実現するために，REMEDY と RegEq[2] というプログラムを使用する．REMEDY については 3 章で説明を行ったとおりである．

RegEq は 2 つの正規表現が与えられたとき，それらが意味的に等価であるか否かを判定するプログラムである．また，等価でない場合，片方の正規表現には受理されるがもう片方の正規表現には受理されない反例を返す．RegEq の等価性判定のアルゴリズムは 2 章で説明を行った [3] ものと同じである．

1.2.2 アルゴリズム

REMEDY と RegEq の 2 つのプログラムを用いて以下のようなアルゴリズムで正規表現を非脆弱化する．

1. (ReDoS 脆弱だと疑われる) 正規表現を REMEDY への入力とする．はじめは positive example，negative example ともに入力として渡さない (つまり positive example の集合，negative example 集合がともに空である).
2. REMEDY から出力された正規表現と元の正規表現を RegEq への入力とし，
   - 等価であるなら修正後の正規表現を返してこのアルゴリズムは終了.
   - 等価でないなら元の正規表現に受理され修正後の正規表現に受理されない文字列を正例に追加し，元の正規表現に受理されず修正後の正規表現に受理される文字列を負例に追加し，それらを修正後の正規表現とともに REMEDY に渡す.
3. 2. を繰り返す.

上記のアルゴリズムを図として表したものを以下に示す.

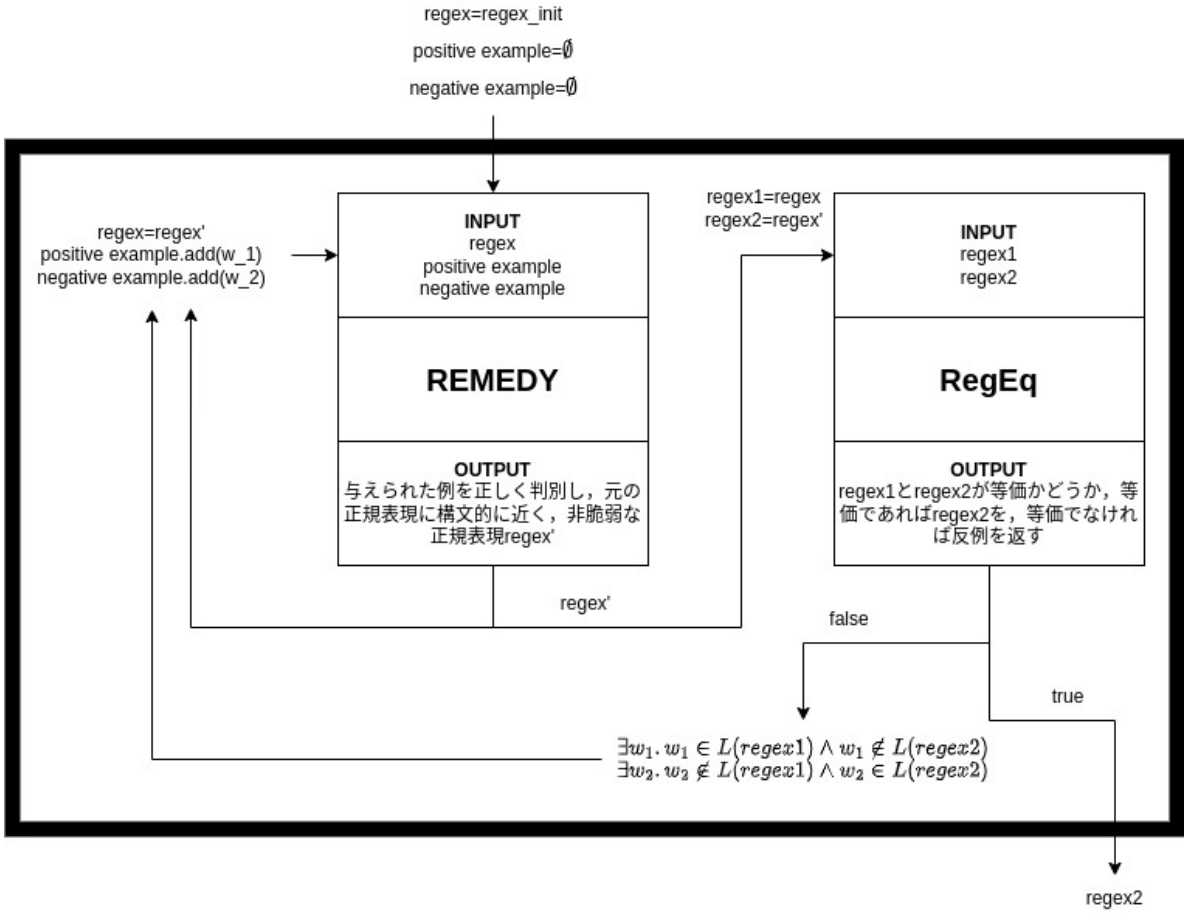

図 1.1　ReDoS 脆弱な正規表現の修正アルゴリズム

このアルゴリズムは停止しない場合がある．それはある正規表現に対してそれと意味的に等価で S1U を満たすようなものが存在しない可能性があるからである．

参考文献

[1] Brink van der Merwe, Nicolaas Weideman, and Martin Berglund. Turning evil regexes harmless. In *Proceedings of the South African Institute of Computer Scientists and Information Technologists*, SAICSIT '17, New York, NY, USA, 2017. Association for Computing Machinery.

[2] Regeq. `https://bakkot.github.io/dfa-lib/regeq.html`. Accessed: 2023-01-21.

[3] John E. Hopcroft and Jeff D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, 1979.