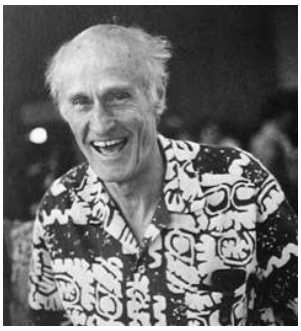
1. INTRODUCTION

History of Regular Expressions

History of Regular Expressions

In 1943, Warren McCulloch (an American cybernetician) and Walter Pitts (an American logician) developed a computer-based model of learning, modelled on how human brains learn. These types of models are generally referred to as artificial neural networks (ANNs), and their specific model is called a McCulloch-Pitts neural network, whose goal was to learn and recognise patterns. In 1951, American mathematician Stephen Cole Kleene created a formal mathematical language to describe these neural networks, and this language eventually evolved into a general pattern matching notation, which we know as Regular Expressions. This notation can be used to explore the structure of any type of text-based patterns.

BIOGRAPHY: Stephen Cole Kleene



Stephen Cole Kleene was born on January 5th, 1909 in Hartford, Connecticut, and died on January 25th, 1994 in Madison, Wisconsin. He is a notable mathematician who helped develop some of the foundations of theoretical computer science (often together with his thesis supervisor, Alonzo Church). He is a founder of the branch of mathematical logic known as recursion theory, and as we know, he invented Regular Expressions in 1951 to describe the McCulloch-Pitts neural net.

The Church–Turing thesis

In the 1930s, Alonzo Church (Kleene's dissertation supervisor) developed a general system of logic, that he called Lambda calculus (λ-calculus), to explore the limits of what it is possible to calculate. At around the same time, British mathematician, Alan Mathison Turing, was working on the same problem using his computation model, that later became known as the "Turing Machine", and he identified an approach that would also help explore the limits of what can be calculated, or computed. In 1952, Stephen Cole Kleene published "Introduction to Metamathematics" where he showed that Lambda calculus and Turing Machines are strictly equivalent, and that they are also equivalent to a third approach by Kurt Gödel called "Recursive Functions". He called this the "Church–Turing" thesis, and it serves as a foundational principle in computer science and helps to establish the limits of computability.

The Church–Turing thesis is part of the broader theory of Computation, which looks at the general question of whether a problem can be solved by a computer; and if it can be solved, is it an approximate solution or a very precise one? We can express these questions in a mathematical language, and when looking at problems that examine pattern matching, we can use Regular Expressions to represent them.