Compiler: Note #1

ant-hengxin

0130

Big Picture:

we are used to expressing lexical rules in regular expressions, but we are writing the lexical analyzer through a `DFA`. That means that there exists some way to convert from regular expressions to `DFA`.

$$\texttt{re} \to \texttt{DFA}$$

But this approach is overly complicated, and for simplicity's sake, let's do this thing in multiple steps:

$$\texttt{re} \xrightarrow[\text{Construction}]{\text{Thompson}} \texttt{NFA} \xrightarrow[\text{Construction}]{\text{Subset}} \texttt{DFA}$$

And we also want to know how to convert a `DFA` to a `re`.

Alphabet Σ: A set of finite symbols.

String `s` over alphabet Σ: A sequence of symbols from Σ. A special string ϵ is the empty string with the property $|\epsilon| = 0$.

String operation: concatenation: `x = dog, y = house` $\Rightarrow$ `xy = doghouse`.

Language L over alphabet Σ: A countable set of strings over Σ.

Language operation:

Suppose `L, M` are languages, we can use set operations to construct new languages:

1. $\texttt{L} \cup \texttt{M:= \{ s | s} \in \texttt{L or s} \in \texttt{M \}}$
2. $\texttt{LM:= \{ s} \in \texttt{L and s} \in \texttt{M \}}$
3. $\texttt{L}^{*}\texttt{:=} \bigcup_{i=0}^{\infty} \texttt{L}^{i}$ `(Kleene closure)`
4. $\texttt{L}^{+}\texttt{:=} \bigcup_{i=1}^{\infty} \texttt{L}^{i}$ `(Positive closure)`

After know what is language, let's consider the regular expression.

Regular expression:

A regular expression over alphabet Σ is defined as follows:

1. ϵ is a regular expression.
2. $\forall$ `a` $\in \Sigma$, `a` is a regular expression.
3. If `r` is a regular expression, then (`r`) is also a regular expression.
4. If `r, s` are regular expressions, then `r|s, rs, r*` are also regular expressions.