在这里我们假设你已经对latex有着一定的了解,并且已经在电脑中装上了相应的texlive和中文字体;
匹配和转录的算法通常遵循所提供的模式或模板的递归结构。匹配使用模式p来实施与tokens tree序列s的等同性，同时将变量x添加到模式环境Θ中。Transcribe使用模板t来生成新的语法s，方法是用基于Θ的替换替换所有的自由模式变量x。

$$match : p * s * \theta \rightarrow (\theta, s); transcribe : t * \theta \rightarrow s \tag{1}$$

```
macro  unless  {
        rule{ $x $y }=>{ if ( ! $x ) $y }
}
unless ( success ) fail() ;
```

下面给出一个代码，显示了一个简单的匹配和转录的例子，作为宏扩展的一部分。

$$\begin{aligned}
&expand_{\sum} \ (unless \cdot (success) \quad fail();) \\
&\hookrightarrow match(x \cdot y, (success) fail(); \qquad \emptyset) \\
&\rightarrow match(y, fail();, \qquad [x \mapsto (success)]) \\
&\rightarrow match(\epsilon, ;, \qquad [x \mapsto (success), y \mapsto fail()]) \\
&\hookrightarrow transcribe(if(!x) \cdot y, \qquad [x \mapsto (success), y \mapsto fail()]) \\
&\rightarrow if(!(success)) \quad fail();
\end{aligned}$$

完整的算法则如下所示。

$$\begin{array}{rlll}
k, n & \text{Token} & h ::= k \mid \{s\} & \text{Token tree} \\
 & & s, r ::= k \cdot s \mid \{s\} \cdot s \mid \epsilon & \text{Syntax Sequence} \\
x & \text{Pattern Variable} & p, q, t ::= k \cdot p \mid \{p\} \cdot p \mid x \cdot p \mid \epsilon & \text{Pattern/Template} \\
\Theta : x \to h & \text{Bound variables} & \Sigma : (n, p, t)^* & \text{Macro Environment}
\end{array}$$

$$\begin{array}{lll}
\text{expand}_\Sigma \ : \ s \to \ s & & \\
\text{expand}_\Sigma \ (k \cdot s) & \hat{=} \ \text{let } (n, p, t) \ = \ \text{first in } \Sigma \text{ s.t. } k = n \wedge \ \text{match } (p, s, \varnothing) & \\
 & \qquad (\Theta, r) \ = \ \text{match } (p, s, \varnothing) & \\
 & \quad \text{in } \ \text{expand}_\Sigma(\text{transcribe}(t, \Theta) \cdot r) & \\
\text{expand}_\Sigma \ (k \cdot s) & \hat{=} \ k \ \cdot \ \text{expand}_\Sigma \ (s) & (\text{otherwise}) \\
\text{expand}_\Sigma \ (\{s\} \cdot s') & \hat{=} \ \{ \ \text{expand}_\Sigma \ (s) \ \} \ \cdot \ \text{expand}_\Sigma \ (s') & \\
\text{expand}_\Sigma \ (\epsilon) & \hat{=} \ \epsilon &
\end{array}$$

$$\begin{array}{ll}
\text{macrofy}_\Sigma \ : s \to s^* & \\
\text{macrofy}_\Sigma \ (h \cdot s) & \hat{=} \{h \cdot r \mid r \in \ \text{macrofy}_\Sigma \ (s)\} \\
 & \cup \ \{\{r'\} \cdot s \mid h = \{s'\} \ \wedge \ r' \in \ \text{macrofy}_\Sigma \ (s')\} \\
 & \cup \ \{n \ \cdot \ \text{transcribe } (p, \Theta) \cdot r' \ \mid \\
 & \qquad (n, p, t) \in \Sigma \ \wedge \ \text{match } (t, h \cdot s, \varnothing)) = (\Theta, r')\} \\
\text{macrofy}_\Sigma \ (\epsilon) & \hat{=} \ \varnothing
\end{array}$$

$$\begin{array}{lll}
\text{match } : p \times s \times \Theta \to (\Theta, s) & & \\
\text{match } (k \cdot p, k' \cdot s, \Theta) & \hat{=} \ \text{match } (p, s, \Theta) & (k = k') \\
\text{match } (\{q\} \cdot p, \{r\} \cdot s, \Theta) & \hat{=} \ \text{match } (p, s, \Theta') & (\text{match } (q, r, \Theta) = (\Theta', \epsilon)) \\
\text{match } (x \cdot p, h \cdot s, \Theta) & \hat{=} \ \text{match } (p, s, \Theta[x \mapsto h]) & \\
\text{match } (\epsilon, s, \Theta) & \hat{=} \ (\Theta, s) &
\end{array}$$

$$\begin{array}{ll}
\text{transcribe } : t \times \Theta \to s & \\
\text{transcribe } (k \cdot t, \Theta) & \hat{=} \ k \ \cdot \ \text{transcribe } (t, \Theta) \\
\text{transcribe } (\{t\} \cdot t', \Theta) & \hat{=} \ \{ \ \text{transcribe } (t, \Theta) \ \} \ \cdot \ \text{transcribe } (t', \Theta) \\
\text{transcribe } (x \cdot t, \Theta) & \hat{=} \ \Theta(x) \ \cdot \ \text{transcribe } (t, \Theta) \\
\text{transcribe } (\epsilon, \Theta) & \hat{=} \ \epsilon
\end{array}$$