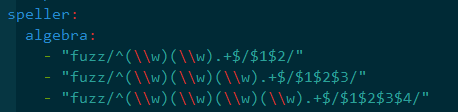
我这儿的码表中就是

减法 sjh st

没法 sj st

码表中没有「减法 sjst」这个编码，只是定义了fuzz拼写运算，输入「sj st」也能打出「减法」这个词。



第一行其实就是定义了每个字取两码的2+2+2的打法

依此类推，第二行就是定义了整句输入时，每字取前三码的打法  
在实际使用中可以混合的，一个字打前两码，另一个字打前三码，  
在整句中都可以  
这样的好处是不会把码表弄乱，  
可以按2+2+2的打法来整句输入，但不会生成2+2+2编码的用户词，码表不会搞乱

比如「中州韵」 os ss ips，这个词码表中只有这一个编码，但你仍然可以通过「os ss ip」打出「中州韵」

这个fuzz就是因为山人的这个特性，向佛振提建议引入的

————————

\* 派生／Derivation : 若對拼寫做正則模式匹配、替換而獲得了新的拼寫，則有效拼寫集合同時包含派生前後的拼寫；否則僅保留原拼寫。  
  
 格式：derive/<模式>/<替換式>/  
 實例一：算式 derive/^([nl])ue$/$1ve/  運算元 nue  結果 nve  
 效果：輸入nve或nue(lve或lue)均可獲得源碼表中與編碼nue(lue)對應的候選  
  
 實例二：算式 derive/^[nl](.\*)$/l$1/  運算元 na  結果 la  
 效果：輸入la可獲得源碼表中與編碼na、la對應的候選；輸入na，候選仍爲碼表中編碼爲na的候選  
  
\* 模糊／Fuzzing : 執行派生運算；派生出的拼寫將獲得「模糊」屬性，可設定將其用作構成詞組的簡碼、但不用於輸入單字。  
  
 格式：fuzz/<模式>/<替換式>/  
 實例：算式 fuzz/^([a-z]).+([a-z])$/$1$2/  
 效果：以首、尾碼爲多字母音節碼的構詞碼。  
     註：需配合 script\_translator 的選項 `translator/strict\_spelling: true` 方可限定該拼寫不用於輸入單字。  
  
\* 縮略／Abbreviation : 執行派生運算；派生出的拼寫將獲得「縮略」屬性，會在音節切分時與通常的拼寫做區分處理。  
  
 格式：abbrev/<模式>/<替換式>/  
 實例：算式 abbrev/^([a-z]).+$/$1/  
 效果：以首字母爲多字母音節碼的縮寫。  
之前使用derive（派生）、abbrev（缩略）两种拼写运算来处理山人方案，效果都不理想  
fuzz的好处就在于  
派生出的拼寫將獲得「模糊」屬性，可設定將其用作構成詞組的簡碼、但不用於輸入單字。

fuzz定义的2+2+2这种打法只用于整句，单字不受干扰，而不像abbrev那样，会使单字的编码也变得混乱  
比如  
呢 oe  
呀 oey  
使用了abbrev的话，打oe，所有以「oe」开头的单字都错乱了…  
完全无法使用