Chinese character synthesis using METAPOST

Candy L. K. Yiu, Wai Wong
Department of Computer Science,
Hong Kong Baptist University
[candyyiu,wwong]@comp.hkbu.edu.hk

Abstract

A serious problem in Chinese information exchange in this rapidly advancing Internet time is the sheer quantity of characters. Commonly used character encoding systems cannot include all characters, and often fonts do not contain all characters either. In professional and scholarly documents, these unencoded characters are quite common. This situation hinders the development of information exchange because special care has to be taken to handle these characters, such as embedding the character as an image. This paper describes our attempt towards solving the problem. Our approach utilizes the intrinsic characteristic of Chinese characters, that is, each character is formed by combining strokes and radicals. We defined a Chinese character description language named *HanGlyph*, to capture the topological relation of the strokes in a character. We are developing a Chinese Character Synthesis System CCSS which transforms *HanGlyph* descriptions into graphical representations. A large part of the CCSS is implemented in METAPOST.

1 Introduction

The rapid advancement of the Internet and the Web provides an effective means of information exchange. However, there is a very serious problem in exchanging Chinese documents: the number of Chinese characters that now exist or have ever existed is unknown. Furthermore, new characters are continually being created. Therefore, no character set can encode *all* Chinese characters.

Even if a character set could encode all Chinese characters, it is very expensive to create Chinese fonts using typical methods and a fairly large number of Chinese characters would be so rarely used that the expense would be very difficult to justify.

One possible solution to this problem is to create an unencoded character according to its composition of strokes and radicals. Several experiments along this line were attempted in the past, but none were very successful. The key reason is that the composition of the strokes and radicals is very complex, and the previous attempts did not effectively divide and resolve the complexity. The section on related works gives a brief survey of some previous attempts.

Our approach to Chinese character synthesis resolves the complexity in two ways. First, we defined a high-level Chinese character description language, *HanGlyph*. It captures the abstract and topological relation of the strokes. Thus, the character description is compact and can be targeted to a variety of rendering styles. The section on the *HanGlyph* Chinese character description language describes the language in more detail. Secondly, we use METAPOST as our rendering engine to take advantage of its meta-ness and the ability of specifying paths and solving linear equations.

The *HanGlyph* language is defined based on many studies of Chinese characters. The section on the sructure of Chinese characters explains the basic structure of Chinese characters for the benefit of readers who are not familiar with them. *HanGlyph* defines 41 basic strokes, 5 operators and a set of relations. A character is built by combining strokes using the operators recursively. *HanGlyph* allows the user to define macros to represent a stroke cluster which can then be re-used in building more complex characters.

The CCSS (stands for Chinese Character Synthesis System) takes *HanGlyph* expressions and renders the characters. It can be divided into three parts: a front-end to translate *HanGlyph* expressions into METAPOST programs, a set of primitive strokes, and a library of METAPOST macros to implement the operators and relations. By varying the parameters to these macros, or redefining the basic stroke macros, Chinese characters in different styles can be formed. Thus, it can create a variety of different fonts from the same *HanGlyph* description.

2 The structure of Chinese characters

Chinese characters, or *hanzi*, have their roots in a very long history. A large body of literature on the study of the written form of Chinese language, dating from as early as more than 2000 years ago (during the *Han* dynasty) up to now, is available. The written form of Indo-European languages consists of around 30 characters. Words are formed using these characters in a linear fashion. In contrast, written Chinese language is denoted by tens of thousands of *hanzi*. The exact number of *hanzi* that have ever existed can never be known.

Many studies have pointed out that each Chinese character is composed from strokes. The number of strokes in a character varies from one for the simplest, up to around 50 for the most complex. Unlike the linear composition of words from characters in Indo-European languages, the arrangement of the strokes in *hanzi* is two-dimensional.

According to the convention of writing Chinese characters, a stroke is a continuous movement of the brush over the writing surface without being lifted up. It is commonly agreed that there are five basic strokes: ㇐ (橫 héng[1]), ㇑ (豎 shù), ㇒ (撇 piě), ㇏ (捺 nà) and ㇔ (點 diǎn).

In practice, each of these basic strokes has some variations depending on the position in a character. For example, the stroke ㇒ 撇 can have two variations: ㇀ (平撇 pìngpiě) (as the top stroke in 千) and ㇓ (豎撇 shùpiě) (as the leftmost stroke in 月). In addition, a number of combinations of these basic movements are considered as strokes because they are connected in a natural way in writing. For example, a ㇐ (橫) followed by a ㇒ (撇) is a single stroke ㇇ called (橫折撇 hèngzhépiě). Modern studies of Chinese characters [1, 9] identified a small set of around 40 strokes as the basic elements of *hanzi*.

Although the arrangements of strokes to form a *hanzi* is very complex, there are some rules that guide the formation of characters. Further, some stroke arrangements are relatively stable and appear in many characters. Some of these arrangements are themselves *hanzi*, for example, 日月; some of them are known as *radicals* which are used in Chinese dictionaries to index characters, for example, 彳匚卩. There are some relatively stable arrangements that are not *hanzi* themselves, nor radicals, but appear in many characters. We will use the term *components* to refer to all these kinds of stroke arrangements, while we use a more general term *stroke clusters* to refer to any arrangements of several strokes.

Except for a small number of very simple characters, such as 人, 二, 十, which cannot be divided into component parts, all *hanzi* can be considered as compositions of certain components. The ways of composing *hanzi* from components are known as the *structure* of the character. Many studies, such as [2] and [8], have identified around 10 different types of structures if one considers how to compose a character from only two components. This does not place serious restrictions, because the composition process can be performed recursively. Figure 1 illustrates the commonly used structures.

[1] The word héng following the *hanzi* name of the stroke is in *pinyin*, a phonetic transcription of Chinese characters. We hope these *pinyin* transcriptions can help readers who do not know Chinese to pronounce the names of the strokes.

3 Related works

Based on the studies of Chinese characters, several attempts have been carried out to create *hanzi* from a structural composition approach.

Toshiyuki *et al* [10] proposed a way of describing Chinese characters using sub-patterns. In principle, their method is similar to the approach of *HanGlyph* because the underlying theory of character structure is intrinsic to all Chinese characters.

Dong [11] and Fan [5] reported their work on the development of a Chinese character design system which took a parametric approach to create characters in different styles. Lim and Kim [7] developed a system for designing Oriental character fonts by composing stroke elements.

Inspired by the success of METAFONT [6] in creating latin character fonts, Hobby and Gu [3] attempted to generate Chinese characters of different styles using METAFONT. A small set of strokes were defined in METAFONT. A small set of radicals were then defined as METAFONT macros by using the strokes. Characters can then be specified as METAFONT programs using these macros as building blocks. By varying some parameters governing the shapes of the strokes, fonts of different styles can be generated. However, the research was not conclusive because they only generated fonts with a very small character set (128 characters).

Another attempt similar to Hobby and Gu was done by Hosek [4] who aimed at generating *hanzi* from a small sets of components.

A common theme of the works mentioned is the difficulty of handling the complexity of the structures and the numerousness of characters. Our approach handles the complexity by using an abstract description and a layered CCSS to decompose the complexity into several sub-problems. On the *HanGlyph* level, we consider strokes as abstract objects.

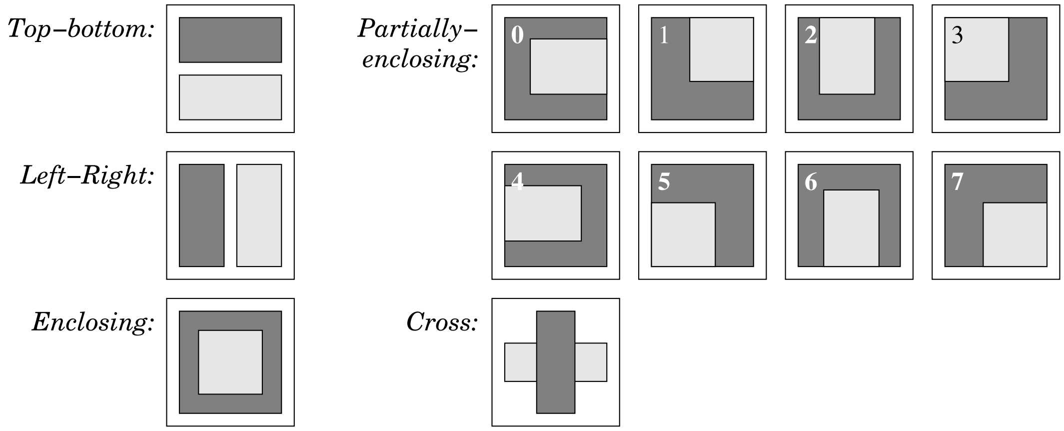


Figure 1: The basic structure of *hanzi*.

We need to specify only the relative positions between these abstract objects. On a lower level, we can work out the outline of the strokes and fine tune the positions.

Another theme of the works mentioned is that they are mainly aimed at the design and generation of character fonts. Our approach can certainly be applied in font generation. However, a very important application area, namely the exchange of Chinese character information, is made possible with our character description language *HanGlyph*.

4 The *HanGlyph* language

Based on the analysis described in previous sections, we defined a Chinese character description language, named *HanGlyph*. The most crucial characteristic of this language is that it is abstract and it captures only the topological relation of the strokes that form a character.

The essential information needed to distinguish a Chinese character is the arrangement of strokes. The precise location of each stroke can vary in a large extent up to a certain threshold, and the character can still be recognized correctly. For example, the following two characters, 士 and 土, comprise exactly the same strokes and in exactly the same arrangement. The only difference between them is the relative length of the two horizontal strokes. Exactly how much longer a horizontal stroke is in these characters is unimportant for distinguishing between them. To recognize the character 土, the threshold is that the upper horizontal stroke must be shorter than the lower one. Therefore, the *HanGlyph* language does not describe the precise geometric information of the characters.

4.1 The strokes

After studying a number of Chinese linguistic and graphological works, we selected a set of 41 strokes as the primitives of *HanGlyph*. Each primitive stroke is assigned a Latin letter as its code so that users can easily write *HanGlyph* expressions using a standard qwerty keyboard. Table 1 lists these primitive strokes.

4.2 The operators and relations

To form a Chinese character, one combines primitive strokes using operators. Five operators are defined as listed in Table 2. Figure 1 illustrates the composition performed by these operators. Each operator combines two operands to form a stroke cluster. This operation continues recursively until the desired character is formed. For example, to describe the character 土, one may first combine two horizontal strokes using the top-bottom operator, then use the cross operator to add a vertical stroke. The *HanGlyph* expression for this character (written in ASCII characters) is `h h=s+`. (*Note*: the expression is in postfix notation.)

However, with only these operators, some characters, like 士 and 土 mentioned above, cannot be distinguished. To resolve situations like this, we can augment the operator with a number of *relation specifiers* to describe the operation in more specific terms. For our sample character 土, the proper *HanGlyph* expression should be `h h=< s+_` where the symbol `<` denotes the relation that the length (i.e., the horizontal dimension) of the upper horizontal strokes must be shorter than the lower one, and the symbol `_` denotes the relation that the two operands of the cross operator, namely 二 and 丨, are aligned at the bottom.

Table 1: *HanGlyph* primitive strokes

Stroke	Name	Code	Examples	Stroke	Name	Code	Examples
㇔	點	`d`	衣主沙	㇒	撇	`p`	大人少
㇔	左點	`D`	心快熱	㇒	平撇	`P`	看千毛
㇏	長點	`f`	不	㇓	豎撇	`q`	用月兒
㇛	撇點	`g`	女好巡	㇜	撇折	`r`	么絲去
㇐	橫	`h`	二三	㇏	捺	`n`	人大丈
㇕	橫折	`i`	口四國	㇏	平捺	`v`	走趕
㇆	橫折鉤	`j`	勾狗月	㇀	提	`t`	刁打地
㇇	橫折撇	`k`	又水冬	㇀	點提	`U`	冰清
㇍	橫折彎	`l`	沿船般	㇅	橫折折	`N`	凹
㇠	橫折彎鉤	`m`	乙吃	㇋	橫折折折	`L`	凸
㇖	橫鉤	`a`	冠皮軍	㇡	橫折折鉤	`J`	万仍
㇑	豎	`s`	十中用	㇊	橫折提	`E`	计
㇗	豎折	`b`	山區忙	㇌	橫撇彎鉤	`K`	隊
㇄	豎彎	`c`	四酒	㇞	豎折折	`B`	鼎亞
㇟	豎彎鉤	`w`	見兒	㇉	豎折折鉤	`C`	馬弓
㇙	豎提	`e`	衣根	㇞	豎折撇	`Q`	专
㇚	豎鉤	`S`	丁寸利	⺄	橫斜鉤	`M`	風颱
㇁	彎鉤	`X`	狗狼家	㇎	橫折折撇	`R`	廷建及
㇂	斜鉤	`Y`	我氣代	㇛	撇點	`z`	学
㇃	臥鉤	`W`	心感	㇇	橫折斜	`F`	子魚矛
㇈	橫豎彎鉤	`o`	九几				

Table 2: *HanGlyph* operators

Name	Symbol	Example
top-bottom 上下	`=`	昌早李
left-right 左右	`|`	明他你
fully enclosing 全包	`@`	回國困
half enclosing 半包†	`^`	凶問區
cross 穿插	`+`	十半木

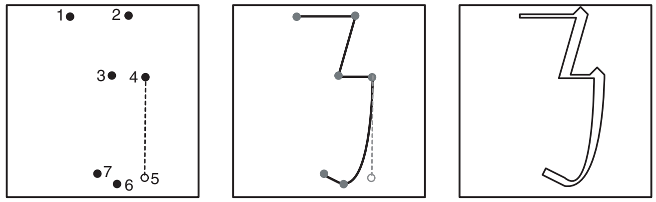
†A digit ranging from `0` to `7` should suffix the half enclose operator to indicate the direction of the opening.

The following four kinds of relations are defined:

1. Dimension — The relations in this group specify the relative dimension of the operands, i.e., comparing the width and height of their bounding boxes. There are four boolean relations: less than (`<`), greater than (`>`), not less than (`!<`), not greater than (`!>`).
2. Alignment — This specifies how the operands are aligned. The possible alignments are at top (`‘`), at bottom (`_`), at left (`[`), at right (`]`) and centered (`#`). More than one alignment can be added to an operation, for example, to align at bottom right (`_]`).
3. Touching — This specifies whether the operands can touch each other. The possible relations are touching (`~`) or not touching (`!~`). When combining two elements to form a new character, the strokes next to the interface of the two elements may or may not touch each other. In general, if the strokes on either side of the interface have the same direction, they will not touch each other, for example, 昌 晶. Otherwise, the strokes may touch each other, for example, 香相.
4. Scale (`/`) — This is used to adjust the width and height of the resulting character after the operation.

4.3 The *HanGlyph* macros

It would be very tedious if every character is described down to all its primitive strokes. It can be seen that certain arrangements of strokes are very common, such as 日 月, and so on. They are used to build up characters. We call them *components*. *HanGlyph* allows *macros* to be defined to stand for a component. For example, the component 日 is a macro with the name `ri_4`. It is defined in terms of



(a) (b) (c)

Figure 2: Basic stroke macros for the stroke ㇋.

another macro `sih` representing the component 口. The actual definitions are as below:

```
let(sih){s i|/h=/}
let(ri_4){sih h@}
```

With a small number of operators and relations, and using a postfix notation, the syntax of the *HanGlyph* language is very simple. This facilitates the development of simple language processors. Figure 5.4 shows the concrete syntax of *HanGlyph*.

After defining the *HanGlyph* language, we have written descriptions of more than 3755 Chinese characters (the first level characters in the GB2312-80 character set). We found that the *HanGlyph* language is adequate for its purpose, to capture the topological relation of the strokes.

5 The CCSS

The purpose of the Chinese Character Synthesis System (CCSS) is to render the *HanGlyph* expressions into a visual representation. For example, the *HanGlyph* expression `h h = <` only specifies that there are two horizontal strokes, one above another, and the upper stroke should be shorter than the lower one. It does not tell us about the exact distance between two strokes. In addition, it does not tell us exactly how much shorter is the upper stroke than the lower one.

The task of the CCSS is to determine and calculate the precise geometric information for each stroke so that a good rendering of characters can be generated.

The CCSS consists of three modules: basic strokes, composition operations and a *HanGlyph*-to-METAPOST translator. The first two modules are implemented as METAPOST macros. The translator is a C program.

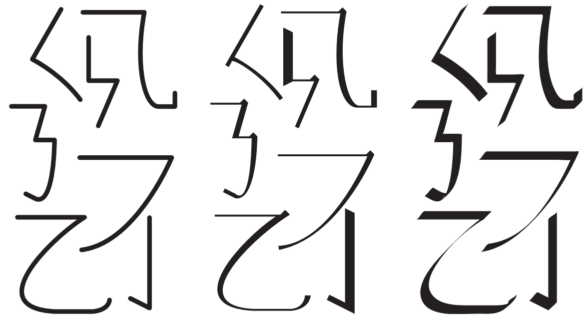
5.1 Basic strokes

Each of the 41 basic strokes listed in Table 1 is implemented as a set of three METAPOST macros:

- A *Control-point* macro defines the control points, handle points and their properties. For example, the stroke ㇋ (横折折鉤 héngzhézhégōu) has six control points and a handle point as shown in Figure 2(a). The locations of the points are specified relative to a reference point. The properties of a control point indicate whether it is a beginning, an end, or a turning point. These properties will be used in creating the outline since its shape at different types of points will be different, for example, the second control point is a turning point, the outline at this point will have a serif shape. The properties will also be used in the composition operations to determine whether certain transformation and positioning opeartions are required.
- A *Skeleton* macro specifies a path passing through the control points. This path is very important. Given two points, the path can be straight or curvy; therefore, this macro traces out the exact stroke skeleton. Figure 2(b) shows the skeletal path of the stroke ㇋.
- An *Outline* macro creates the outline for the stroke. It is defined relative to the control points and the skeletal path. Figure 2(c) shows the outline for the stroke ㇋.

The first reason for organizing the stroke composition into three macros is to avoid distortion. In composition operations, each stroke will be transformed several times before the whole character is formed. If the stroke including the outline is represented in one macro, the transformation will distort the stroke thickness and even the direction in certain slant strokes.

Another reason is to provide meta-ness and flexibility. This organization provides several levels of style changes. The first level is to vary the parameters of the outline macros. For instance, changing the stroke thickness parameter will result in characters of varying stroke thickness. If we change the set of outline macros, we can create completely different font styles, but they may still be recognised as a family because the locations of control points are



(a) (b) (c)

Figure 3: Variations of strokes having the same skeleton.

unchanged. More variations can be achieved if the control point macros and the skeleton macros are also changed. The result will be a completely different font family. Figure 3 shows three variations of the same skeleton. Figure 3(a) is the simple skeletal path of the strokes. Figure 3(b) is the outline with serif. Figure 3(c) is generated by stroking the skeleton with a pen angled at 25 degrees.

5.2 Composition operations

CCSS implements five operations corresponding to the five operators defined in *HanGlyph*. These operations are implemented as METAPOST macros. Again, the operators in *HanGlyph* represent abstract operations, like the *Top-bottom* operator (=) only means 'put an operand on top of another'. It carries no precise geometric information. Given this abstract instruction, the macro implementing this operation has to calculate the exact location and dimension of each operand. The resulting rendering should be a well-balanced and well-positioned arrangement of strokes.

One important task of the composition operation is to estimate the relative sizes and positions for its operands so that the result is visually well-balanced. For example, Figure 4 illustrates two characters having the same radical 木 (mù) on their left side. The width of this radical in the first character 林(lín) is larger than that in the second character 樹 (shù) because the right-hand side of 樹 has many more strokes. We have found that the ratio of the widths of the two components is proportional to the ratio of the sums of the lengths of the strokes and the number of strokes of the components.

HanGlyph expressions may include a number of relations to augment the operators. The composition operations need to calculate the exact dimension and transformation to apply to each operand. For example, the character 人 (rén) is composed of

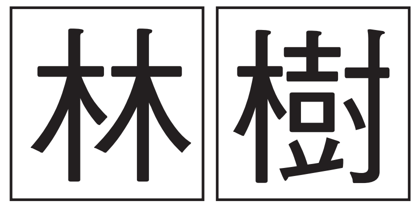


Figure 4: The same radical having different widths.

two strokes where the right-hand one is shorter and the two are aligned at the bottom. The composition operation will first scale the right-hand stroke down to a default size, and then translate it so that the bottom lines of the two strokes are aligned to the bottom of the character box as shown in Figure 5.

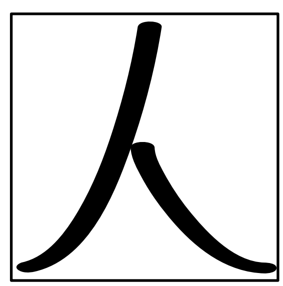


Figure 5: A character composed of two strokes aligned to bottom.

While we are talking about transforming the strokes, in fact, only the control points and the skeletal path are transformed. After all strokes forming a character have been put at the right position, the outline is drawn. This avoids the outline being distorted by the transformations.

5.3 *HanGlyph* to METAPOST translation

The front-end of the CCSS is the translator that converts *HanGlyph* expressions into METAPOST programs. The current implementation of the translator puts each *HanGlyph* expression into a METAPOST figure. Within each figure, the appropriate sequence of composition operation macros is called to render the character. The output of the system is a set of PostScript files.

This implementation provides a simple way to render the *HanGlyph* expressions and obtain previews of the characters. It facilitates the fine-tuning of the composition operations. Future implementations can streamline the process according to the requirements of the target application. For instance, a back-end processor can be added to convert the PostScript output into a particular format, such as a PostScript Type 3 font.

5.4 The syntax of *HanGlyph*

Figure 6 shows the concrete syntax of *HanGlyph* in an augmented BNF notation.

6 Conclusion

This paper describes an attempt to synthesize Chinese characters from an abstract description. A Chinese character description language known as *HanGlyph* has been defined. A Chinese character synthesis system is being developed. It is implemented in METAPOST and C, and the output is rendered in PostScript. The preliminary results show that the approach is very promising. Some of the characters generated by the CCSS are shown in Figure 7.

Currently, we are in the process of fine-tuning the composition parameters. We hope the system is able to produce visually pleasing characters. There are many factors that may affect the quality of the output, for example, the thickness of the strokes, the allocation of the space occupied by each component, and so on. Therefore, a considerable amount of experimentation is required to detemine a set of parameters for composing characters.

There are many applications of such a system. The most important ones are in exchanging Chinese textual information in an open, heterogenous environment, and in Chinese font generation.

References

[1] 蘇培成 (SU Pei Cheng). 二十世紀的現代漢字研究 (*The 20th century research on modern Chinese characters*). 書海出版社 (Su Hai Press), 2001.

[2] 劉連元 (LIU Lian Yuan). 漢字拓扑結構分析 (Analysis of the topological structure of Chinese characters). In 漢字. 上海教育出版社 (Shanghai Education Press), 1993.

[3] John Hobby and Gu Guoan. A Chinese metafont? *TUGboat*, 5(2):119–136, 1984.

[4] Don Hosek. Design of Oriental characters with METAFONT. *TUGboat*, 10(4):499–501, 1989.

[5] Fan Jiangping. Towards intelligent Chinese character design. In *Raster Imaging and Digital Typography II (RIDT91)*, pages 166–176. Cambridge University Press, 1992.

[6] Donald E. Knuth. *The METAFONTbook*. Addison-Wesley, 1986.

[7] Soon-Bum Lim and Myung-Soo Kim. Oriental character font design by a structured composition of stroke elements. *Computer-aided design*, 27(3):193–207, 1995.

[8] 傅永和 (FU Yong He). 漢字結構和構造成分的基楚研究 (*Basic research on the structure of Chinese characters and their constituents*). 上海教育出版社 (Shanghai Education Press), 1993.

[9] 傅永和 (FU Yong He). 中文信息處理 (*Chinese information processing*). 廣東教育出版社 (Guangdong Education Press), 1999.

[10] Sakai Toshiyuki, Nagao Makoto, and Terai Hidekazu. A description of Chinese characters using subpatterns. *Information Processing Society of Japan Magazine*, 10:10–14, 1970.

[11] Dong YunMei and Li Kaide. A parametric graphics approach to Chinese font design. In *Raster Imaging and Digital Typography II (RIDT91)*, pages 156–165. Cambridge University Press, 1992.

$$
\begin{array}{rcll}
\langle \mathit{hanglyph} \rangle & ::= & \langle \mathit{expr} \rangle + & (1) \\
\langle \mathit{expr} \rangle & ::= & \langle \mathit{glyph\_expr} \rangle \mid \langle \mathit{macro} \rangle \mid \langle \mathit{char} \rangle & (2) \\
\langle \mathit{glyph\_expr} \rangle & ::= & \langle \mathit{glyph} \rangle \texttt{;} & (3) \\
\langle \mathit{macro} \rangle & ::= & \texttt{let(}\langle \mathit{id} \rangle\texttt{)\{}\langle \mathit{glyph} \rangle\texttt{\}} & (4) \\
\langle \mathit{char} \rangle & ::= & \texttt{char(}\langle \mathit{code} \rangle\texttt{)\{}\langle \mathit{glyph} \rangle\texttt{\}} & (5) \\
\langle \mathit{glyph} \rangle & ::= & \langle \mathit{glyph} \rangle\langle \mathit{glyph} \rangle\langle \mathit{opn} \rangle & (6) \\
 & \mid & \langle \mathit{stroke} \rangle & \\
 & \mid & \langle \mathit{id} \rangle & \\
\langle \mathit{opn} \rangle & ::= & \langle \mathit{parallel\_operator} \rangle\langle \mathit{parallel\_rels} \rangle & (7) \\
 & \mid & \texttt{@}\langle \mathit{full\_enc\_rels} \rangle & \\
 & \mid & \texttt{\^{}}\langle \mathit{dir\_all} \rangle\langle \mathit{half\_enc\_rels} \rangle & \\
 & \mid & \texttt{+}\langle \mathit{cross\_rels} \rangle & \\
\langle \mathit{parallel\_operator} \rangle & ::= & \texttt{=} \mid \texttt{|} & (8) \\
\langle \mathit{dir} \rangle & ::= & \texttt{.}(\texttt{E} \mid \texttt{S} \mid \texttt{W} \mid \texttt{N} \mid \texttt{e} \mid \texttt{s} \mid \texttt{w} \mid \texttt{n}) & (9) \\
\langle \mathit{dir\_all} \rangle & ::= & \langle \mathit{dir} \rangle \mid \texttt{.}(\texttt{NE} \mid \texttt{SE} \mid \texttt{NW} \mid \texttt{SW} \mid \texttt{ne} \mid \texttt{se} \mid \texttt{nw} \mid \texttt{sw}) & (10) \\
\langle \mathit{parallel\_rels} \rangle & ::= & \langle \mathit{dimens} \rangle?\langle \mathit{aligns} \rangle?\langle \mathit{touch} \rangle?\langle \mathit{scale} \rangle? & (11) \\
\langle \mathit{full\_enc\_rels} \rangle & ::= & \langle \mathit{dimens} \rangle?\langle \mathit{touch} \rangle?\langle \mathit{scale} \rangle? & (12) \\
\langle \mathit{half\_enc\_rels} \rangle & ::= & \langle \mathit{dimens} \rangle?\langle \mathit{aligns} \rangle?\langle \mathit{touch} \rangle?\langle \mathit{scale} \rangle? & (13) \\
\langle \mathit{cross\_rels} \rangle & ::= & \langle \mathit{dimens} \rangle?\langle \mathit{align} \rangle? & (14) \\
 & & (\langle \mathit{align} \rangle \mid \langle \mathit{intercept} \rangle)?\langle \mathit{scale} \rangle? & \\
\langle \mathit{intercept} \rangle & ::= & \texttt{*}\langle \mathit{dir} \rangle((\langle \mathit{+int} \rangle(\langle \mathit{real} \rangle?\langle \mathit{int} \rangle?))? & (15) \\
\langle \mathit{dimens} \rangle & ::= & \langle \mathit{comp} \rangle(\langle \mathit{comp} \rangle \mid \langle \mathit{num} \rangle)? & (16) \\
 & \mid & \langle \mathit{num} \rangle\langle \mathit{comp} \rangle? & \\
 & \mid & \langle \mathit{num} \rangle\texttt{,}\langle \mathit{num} \rangle & \\
\langle \mathit{comp} \rangle & ::= & \texttt{<} \mid \texttt{>} \mid \texttt{!<} \mid \texttt{!>} \mid \texttt{-} & (17) \\
\langle \mathit{aligns} \rangle & ::= & \langle \mathit{align} \rangle\langle \mathit{align} \rangle? & (18) \\
\langle \mathit{align} \rangle & ::= & \texttt{`} \mid \texttt{\_} \mid \texttt{[} \mid \texttt{]} \mid \texttt{\#} & (19) \\
\langle \mathit{touch} \rangle & ::= & \texttt{\~{}}\langle \mathit{dir\_spec} \rangle * & (20) \\
 & \mid & \texttt{!\~{}}(\langle \mathit{dir\_spec} \rangle\langle \mathit{num} \rangle?) * & \\
\langle \mathit{dir\_spec} \rangle & ::= & (\texttt{.}\langle \mathit{dir} \rangle) + & (21) \\
\langle \mathit{scale} \rangle & ::= & \texttt{/}\langle \mathit{num} \rangle? & (22) \\
\langle \mathit{num} \rangle & ::= & \langle \mathit{int} \rangle \mid \langle \mathit{real} \rangle & (23)
\end{array}
$$

Figure 6: The syntax of *HanGlyph* descriptions.

Figure 7: Some characters generated by CCSS.