H.264(二)语法函数、类别和描述子的规范

语法函数

以下函数用于语法描述。这些函数假定解码器中存在一个比特流指针，这个指针指向比特流中解码过程要读
取的下一个比特的位置。
byte_aligned( ) 的规定如下：

- 如果比特流的当前位置是在字节边界，即，比特流中的下一比特是字节第一个比特，byte_aligned( )的返回值为TRUE。
- 否则，byte_aligned( )的返回值为FALSE。

more_data_in_byte_stream( )，只有在附件 B 规定的字节流 NAL 单元语法结构中使用，规定如下：

- 如果字节流中后续还有更多数据，more_data_in_byte_stream( )的返回值为TRUE。
- 否则，more_data_in_byte_stream( ) 的返回值为 FALSE。

more_rbsp_data( ) 的规定如下：

- 如果在rbsp_trailing_bits( )之前的RBSP中有更多数据，more_rbsp_data( ) 的返回值为TRUE。
- 否则，more_rbsp_data( )的返回值为 FALSE。

判断 RBSP 中是否有更多数据的方法由应用规定（或者附件 B 中使用字节流格式的应用）。
more_rbsp_trailing_data( ) 的规定如下：

- 如果RBSP中有更多数据，more_rbsp_trailing_data( ) 的返回值为TRUE。
- 否则，more_rbsp_trailing_data( ) 的返回值为 FALSE。

next_bits(n)提供比特流中接下来的比特用于比较的目的，而不需要移动比特流指针。该函数使比特流中的下n 个比特可见，n 在这里是函数的参数。当用在附件 B 规定的字节流中时，如果剩余的字节流已不足 n 个比特，next_bits(n)返回值为 0。

read_bits(n) 从比特流中读取下面的 n 个比特，并且将比特流指针向前移动 n 个比特。当 n 等于 0 时，
read_bits(n)的返回值为 0 并且不移动比特流指针。

类别

类别（在表中以 C 表示）规定条带数据可以至多划分为三种条带数据类别。

条带数据类别 A 包含了类别 2的所有语法元素。条带数据类别 B 包含了类别 3 的所有语法元素。条带数据类别 C 包含了类别 4 的所有语法元素。

其他类别值的含义不作规定。某些语法元素需要使用两个类别值，这两个值通过竖线分开。在这些情况下，本文将会进一步说明应用的类别值的含义。对于在其他语法结构中使用的语法结构，它所包含的所有语法元素的类别值都应列出，通过竖线来分开。

如果语法元素或者语法结构的类别标为”All”，它可以出现在所有的语法结构中。对于用在其他语法结构中的语法结构，语法表格中的数字类别值如果处于包含了一个类别值为“All”的语法结构中，那么该数字类别值被认为能够应用到类别为“All”的语法元素值。

描述子

描述子是指从比特流提取句法元素的方法，即句法元素的解码算法，每个句法元素都有相对应的描述子。由于 H.264 编码的最后一步是熵编码，所以这里的描述子大多是熵编码的解码算法。H.264定义了如下几种描述子：

— ae(v)：上下文自适应算术熵编码语法元素。该描述符的解析过程在9.3节中规定。

— b(8)：任意形式的8比特字节。该描述符的解析过程通过函数read_bits(8)的返回值来规定。

— ce(v)：左位在先的上下文自适应可变长度熵编码语法元素。该描述符的解析过程在9.2节中规定。

— f(n)：n位固定模式比特串（由左至右），左位在先， 该描述符的解析过程通过函数read_bits(n)的返回值来规定。

— i(n)：使用n比特的有符号整数。在语法表中，如果*n*是‘v’，其比特数由其它语法元素值确定。解析过程由函数read_bits(*n*)的返回值规定，该返回值用最高有效位在前的2的补码表示。

— me(v)：映射的指数哥伦布码编码的语法元素，左位在先。解析过程在9.1中定义。

— se(v)：有符号整数指数哥伦布码编码的语法元素位在先。解析过程在9.1中定义。

— te(v)：舍位指数哥伦布码编码语法元素，左位在先。解析过程在9.1中定义。

— u(n)：*n*位无符号整数。在语法表中，如果*n*是‘v’，其比特数由其它语法元素值确定。解析过程由函数read_bits(*n*)的返回值规定，该返回值用最高有效位在前的二进制表示。

— ue(v)：无符号整数指数哥伦布码编码的语法元素，左位在先。解析过程在 9.1 中定义。

描述子乍一看很多，其实我们可以把它们分为三类：

（1）连续读取n（包含b(8)）个比特：b(8)、f(n)、i(n)、u(n)，其中只有i(n)为有符号整数，并且几乎用不到，其他情况则顺序从左至右，读取固定数量的n个bit即可。如上面所讲的NALU Header的句法元素

（2）指数哥伦布编码：ue(v)、me(v)、se(v)、te(v)，这四个描述子，都是指数哥伦布编码。注意到它们使用的变量是v而不是n，它们的值，并不是直接等于读取固定长度的比特。而是先根据其他句法元素的值，来确定读取多少比特，然后再将读取到的比特，进行转换才能得到所求句法元素的值。

关于它们，我们后面会单独开几篇来介绍。

（3）CAVLC、CABAC：ae(v)、ce(v)，同指数哥伦布编码一样，CAVLC和CABAC也属于变长编码，这也是我们需要学习的一大重点。

利用描述子解析句法元素

所以这时，只要我们懂得，各个描述子是如何计算的，我们就能根据它们解析出句法元素的值。不过需要注意的是，有时候我们会看到这种情况：

macroblock_layer( ) {	C	Descriptor
mb_type	2	ue(v) | ae(v)
if(mb_type == I_PCM) {		
while(!byte_aligned( ))		
pcm_alignment_zero_bit	3	f(1)
for(i = 0; i < 256; i++)		
pcm_sample_luma[i]	3	u(v)
for(i = 0; i < 2 * MbWidthC * MbHeightC; i++)		
pcm_sample_chroma[i]	3	u(v)

可以看到，在解析宏块层mb_type的时候，该句法元素对应了两个描述子，分别为ue(v)和ae(v)，并且它们之间用竖线 “|” 分隔开。

H.264协议规定，出现这种情况，得根据另一句法元素entropy_coding_mode_flag 的值来判断：

如果entropy_coding_mode_flag等于0，则使用左边的描述子，这时为ue(v)。

如果entropy_coding_mode_flag等于1，则使用右边的描述子，这时为ae(v)。

为什么描述子是正确打开码流解析的第一步

我们如果要进行码流解析，第一步则是进行句法元素的解析，而句法元素的解析，又依赖于刚才所讲的那几种描述子。通过刚才的学习我们也知道，学习描述子，其实就相当于学习熵编码。

所以我们接下来，就从学习描述子开始。