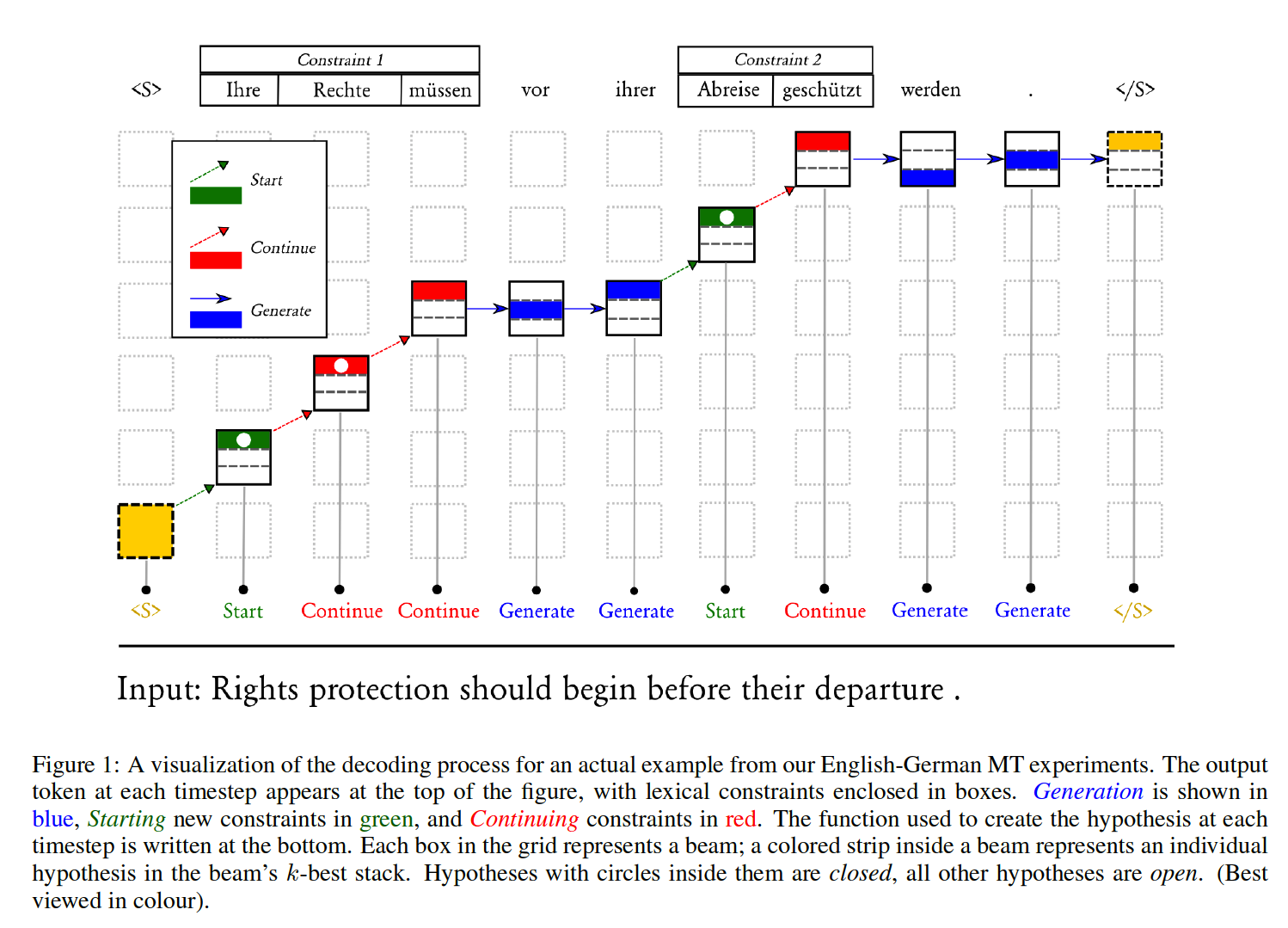
Grid Beam Search

这个方法来源于论文《Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search》，grid beam search[GBS]也就是网格beam search，要求在输出中包含某些特定的词或者短语。限制也可以分为两种，词限制和短语限制。

下面是GBS的一个大致流程。



其中，可以看到分为三种情况：

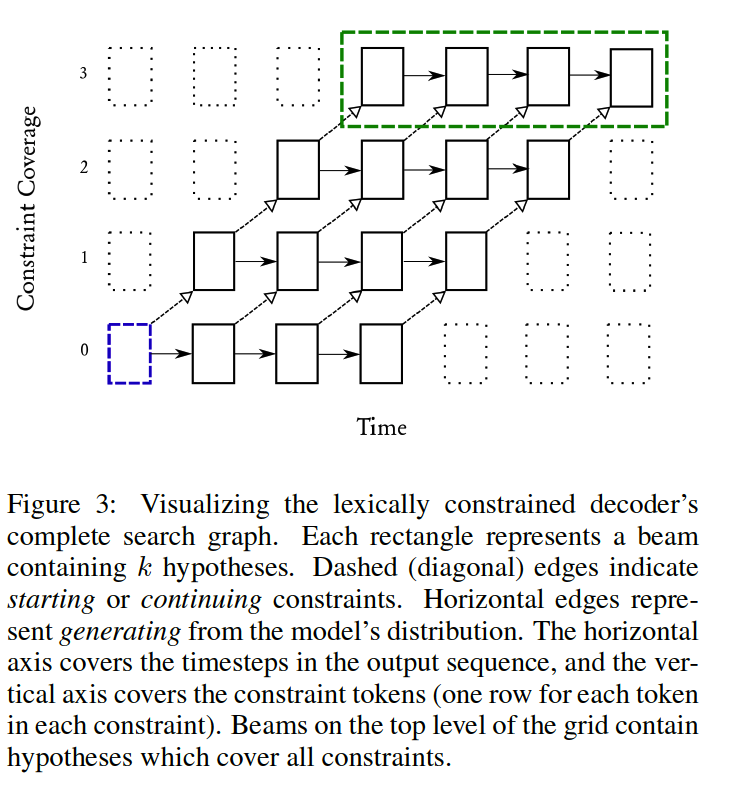
* start：constraint的起始状态。
* continue：继续生成constraint，这暗示它是一个短语级别的限制的中间状态。
* generate：从模型分布选出的词不是constrain

作者对于模型进行beam search的过程中产生的hypothesis分为了两类：

* Open hypothesis：可以从模型产生的输出分布中获取下一个token，这种情况其实可以分为两种，一种是选取的token不在constraints中，一种是选取的token是某个constraint的起始状态。
* Closed hypothesis：这个hypothesis下一步生成的token，是由这个未完成的constraint决定的。

所以，我对于这个限制某些词生成的过程的理解，其实也是按照正常的beam search过程进行，只是在遇到满足某个限制时，就看他是word级别限制还是短语级别限制来决定下一步生成token的方式【是从模型产生的分布中选取，还是强制只能生成某个词】，最终只从满足所有限制条件的hypothesis中选取分值最高的作为结果。

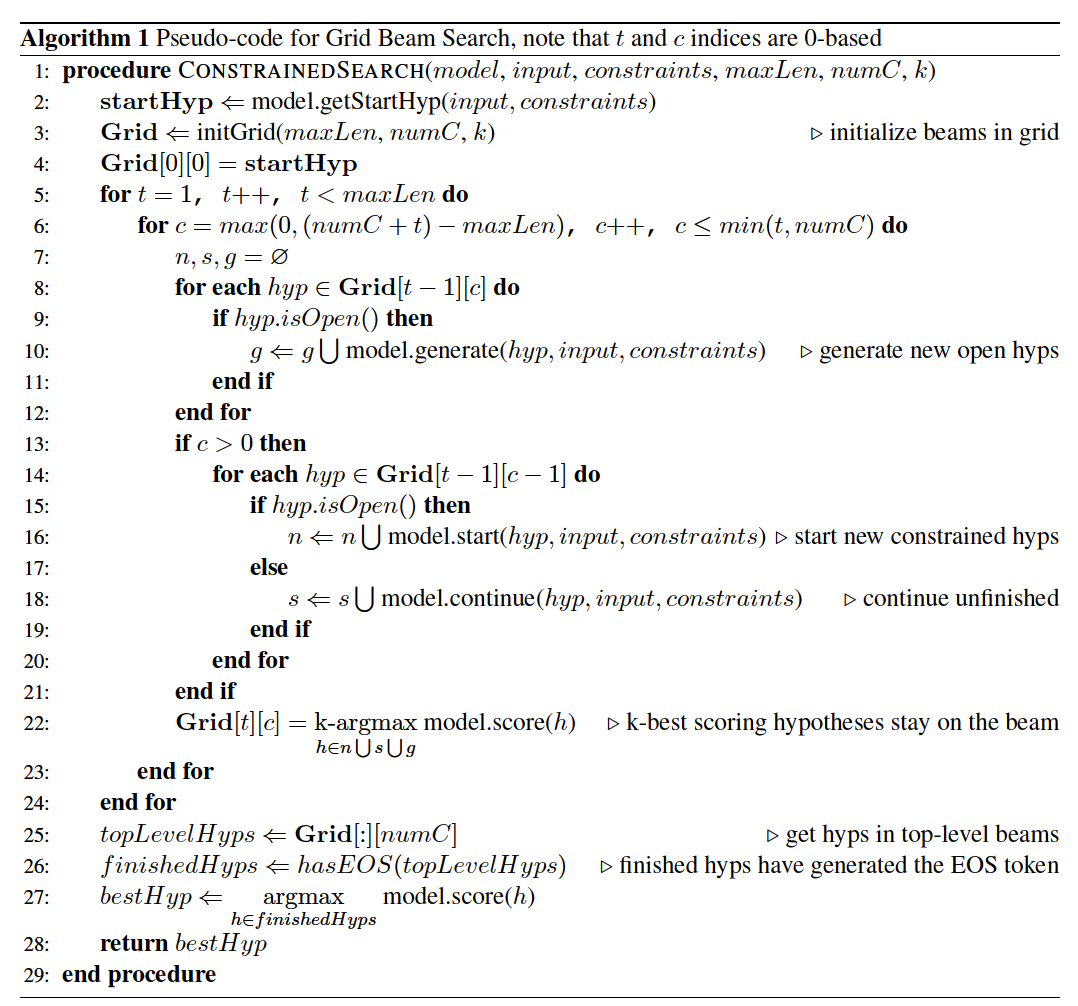
下面是GBS的可视化过程：



其中横轴是时刻t，纵轴是constraint数目。其中可以看到，G[t][c]只有两个来源，

* 一个是G[t-1][c]：表明在t时刻生成的token是从模型输出中选择的，并且该token不是某一个constraint
* 一个是G[t-1][c-1]：表明在t时刻生成的token是一个constraint，但是它可以是从模型输出中选择的【对应图1中的start状态】，也可以是位于短语级别限制的中间状态【对应图1中的continue状态】

下面是整个过程的伪代码：



所以模型在生成句子时，其实是不知道这些constraint的存在的，但是我们可以通过一些特征来表示当前的hypothesis是否包含constraint。

然后对于图3中的网格最上方的那些hypothesis就是满足了所有constraint的，所以当他们生成了EOS符号后，就可以加入到候选集中，最后选择分数最高的作为答案。