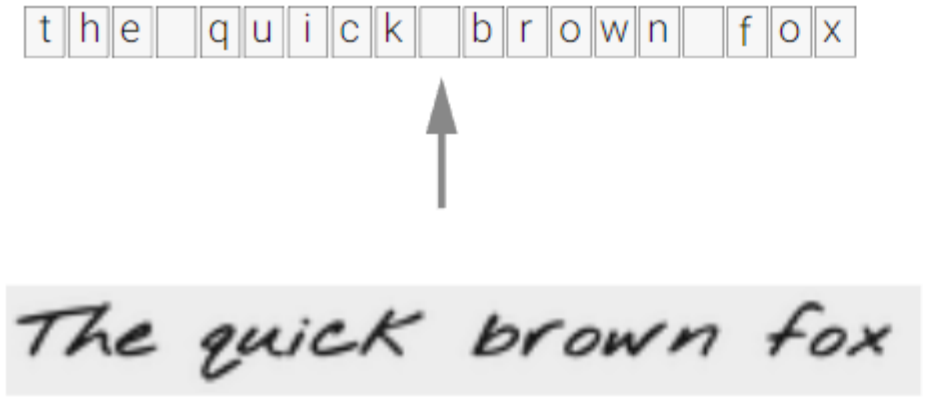


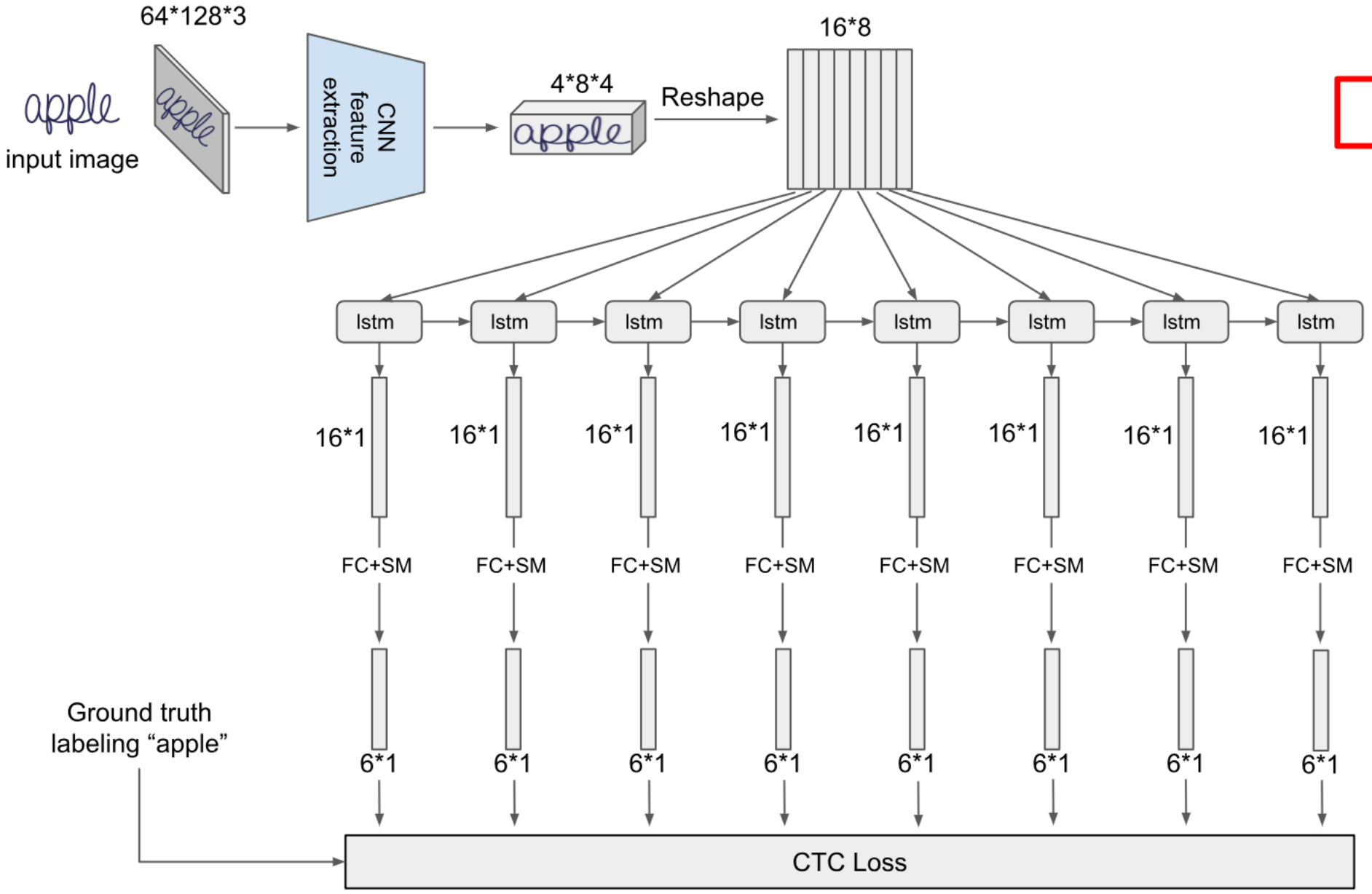
Handwriting recognition: The input can be (x, y) coordinates of a pen stroke or pixels in an image.

Speech recognition: The input can be a spectrogram or some other frequency based feature extractor.

Connectionist Temporal Classification Labelling Unsegmented Sequence Data with Recurrent Neural Networks

主讲人：杜臣

2018.07.14



Link to CTC Loss paper

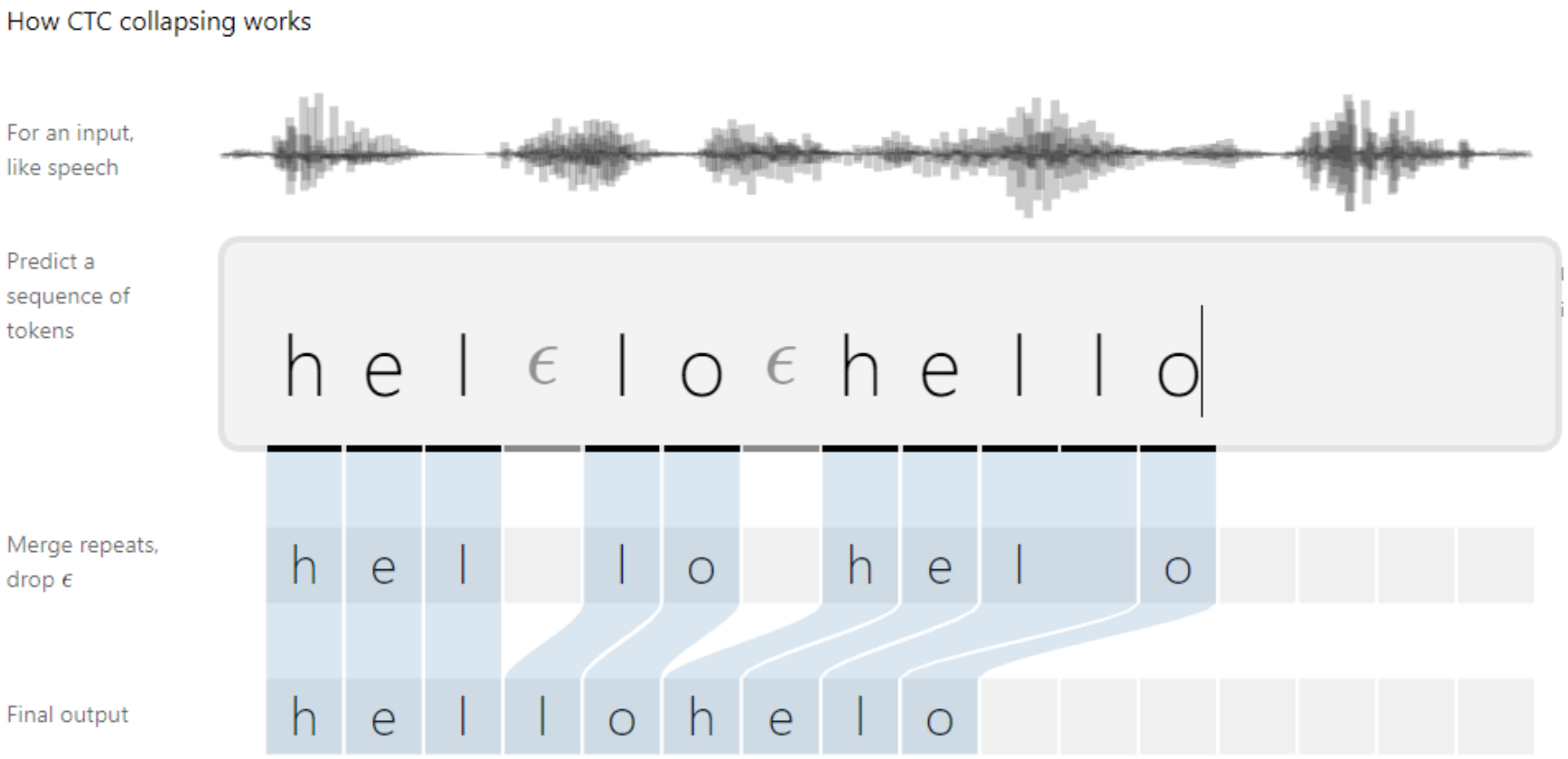
1、CTC loss用来处理输入特征和label不对齐的训练问题。

2、label长度需小于输入长度。

blank“ _”的作用

1、CTC引入了blank标记”_”，用来表示当前输出为空，以及用来分割连续的相同输出。

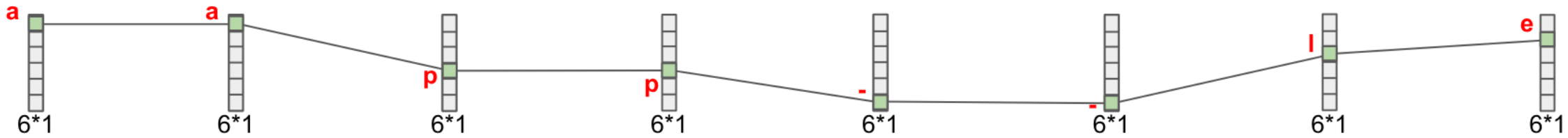
2、解码时先去除重复字符，再去除“_”



Path1: “ap-pl-ee” B(“ap-pl-ee”) Labeling: “apple”

$$p(\text{“}ap-pl-ee\text{”}) = y_a^1 \cdot y_p^2 \cdot y_-^3 \cdot y_p^4 \cdot y_l^5 \cdot y_-^6 \cdot y_e^7 \cdot y_e^8$$

从RNN(LSTM)的输出中，能组成很多种路径，每一条路径的概率就是路径中每一时刻输出的概率乘。



Path1: “ap-pl-ee” —B(“ap-pl-ee”)→ Labeling: “apple”

$$p(\text{“}ap-pl-ee\text{”}) = y_a^1 \cdot y_p^2 \cdot y_-^3 \cdot y_p^4 \cdot y_l^5 \cdot y_-^6 \cdot y_e^7 \cdot y_e^8$$

Path2: “aapp--le” —B(“aapp--le”)→ Labeling: “aple”

$$p(\text{“}aapp--le\text{”}) = y_a^1 \cdot y_a^2 \cdot y_p^3 \cdot y_p^4 \cdot y_-^5 \cdot y_-^6 \cdot y_l^7 \cdot y_e^8$$

Path1: “ap-pl-ee” —B(“ap-pl-ee”)→ Labeling: “apple”

$$p(\text{“}ap-pl-ee\text{”}) = y_a^1 \cdot y_p^2 \cdot y_-^3 \cdot y_p^4 \cdot y_l^5 \cdot y_-^6 \cdot y_e^7 \cdot y_e^8$$

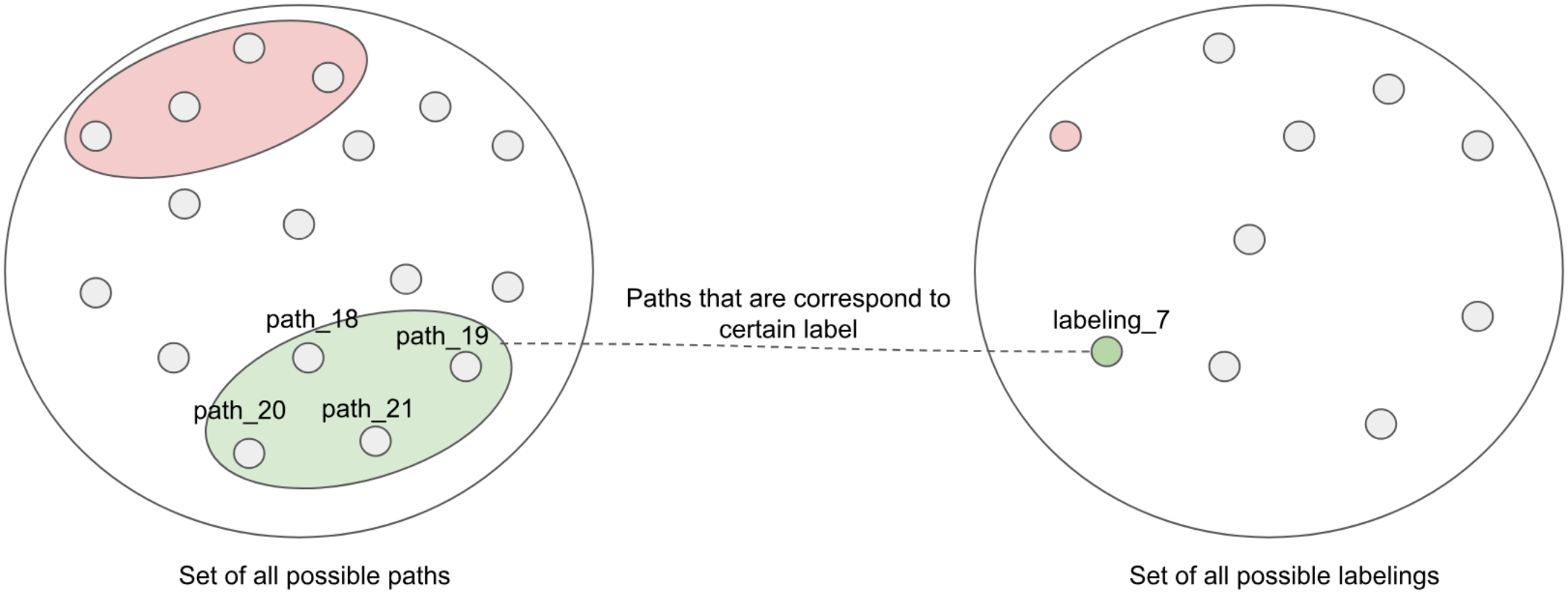
Path2: “aapp--le” —B(“aapp--le”)→ Labeling: “aple”

$$p(\text{“}aapp--le\text{”}) = y_a^1 \cdot y_a^2 \cdot y_p^3 \cdot y_p^4 \cdot y_-^5 \cdot y_-^6 \cdot y_l^7 \cdot y_e^8$$

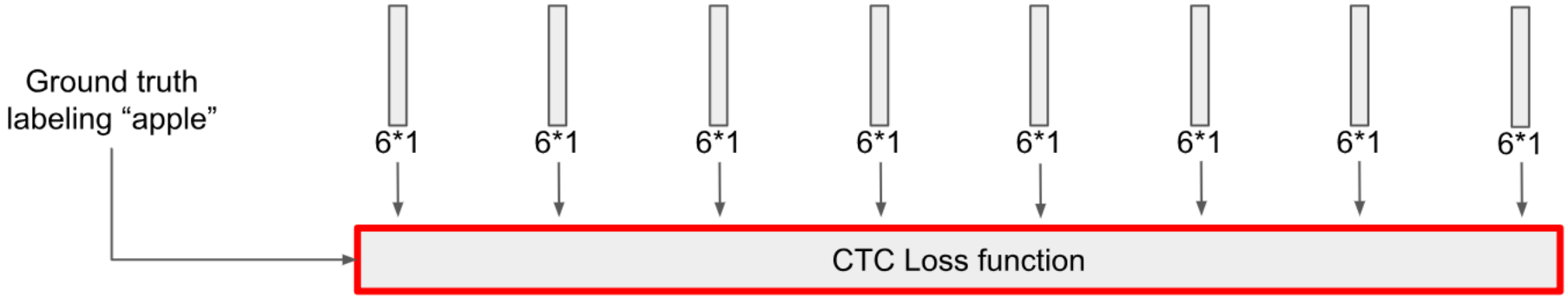
Path3: “aap--lzz” —B(“aap--lzz”)→ Labeling: “aplz”

... ...

PathN: “--z--eee” —B(“--z--eee”)→ Labeling: “ze”



p(labeling_7) = sum of probabilities of all corresponding paths =
= p(path_18) + p(path_19) + p(path_20) + p(path_21)



CTC Loss = -ln(p(“apple”))

It seems very simple, but there is one tricky problem.

In our example there are exist 6^8 = 1 679 616 possible paths. For a larger dictionary size and for a larger number of lstm steps number of possible paths will be huge.

For a given labeling we can not compute the sum of all paths probabilities, because there are very many of these.

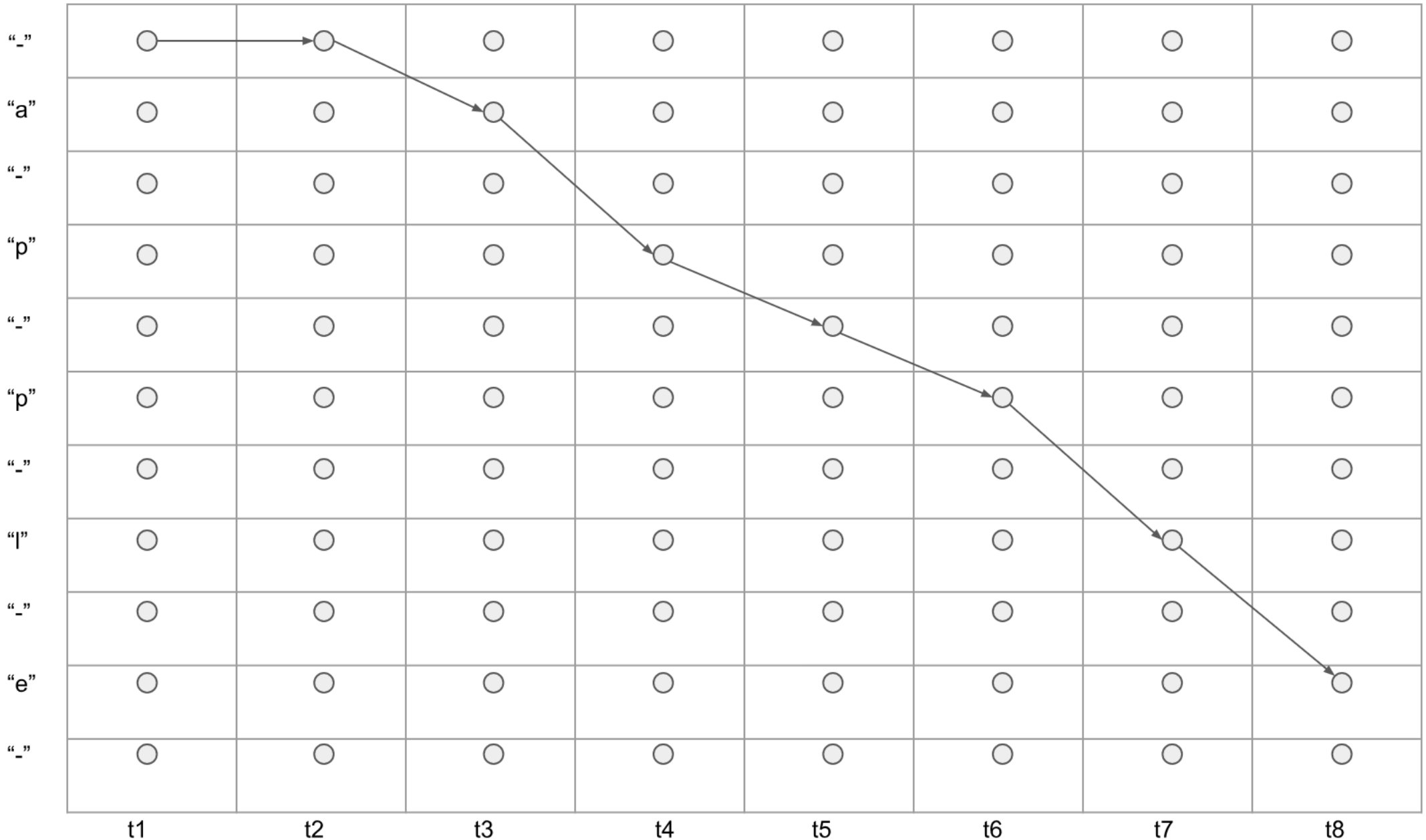
Fortunately there is an efficient way of calculation ground truth labeling probability. The problem can be solved with dynamic programming algorithm.

CTC Loss calculation: dynamic programming algorithm

With this table we can model all paths that are correspond to ground truth labeling “apple”

	t1	t2	t3	t4	t5	t6	t7	t8
“-”	○	○	○	○	○	○	○	○
“a”	○	○	○	○	○	○	○	○
“-”	○	○	○	○	○	○	○	○
“p”	○	○	○	○	○	○	○	○
“-”	○	○	○	○	○	○	○	○
“p”	○	○	○	○	○	○	○	○
“-”	○	○	○	○	○	○	○	○
“l”	○	○	○	○	○	○	○	○
“-”	○	○	○	○	○	○	○	○
“e”	○	○	○	○	○	○	○	○
“-”	○	○	○	○	○	○	○	○

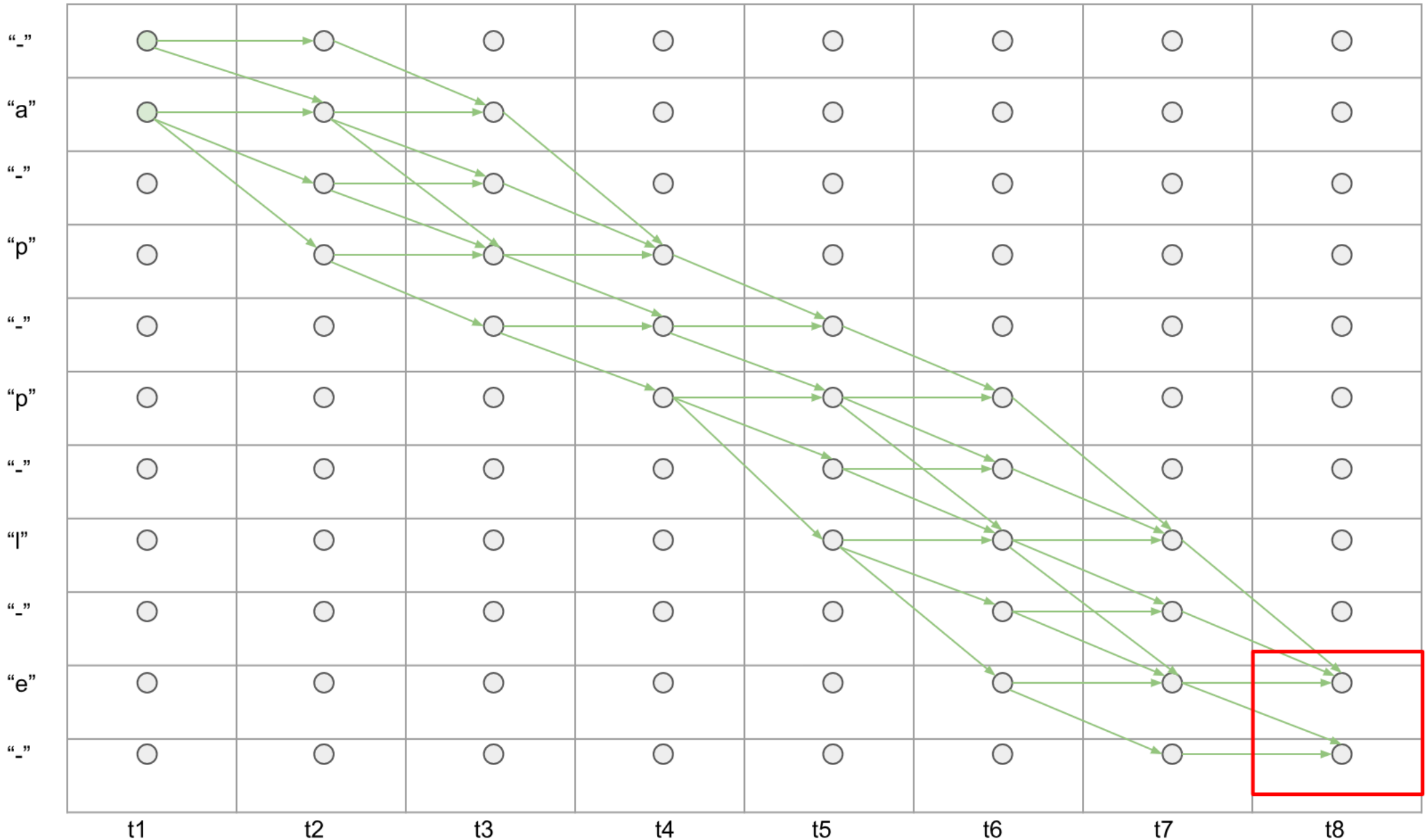
For example: path “--ap-ple” can be mapped to labeling “apple”. i.e: B(“--ap-ple”) = “apple”



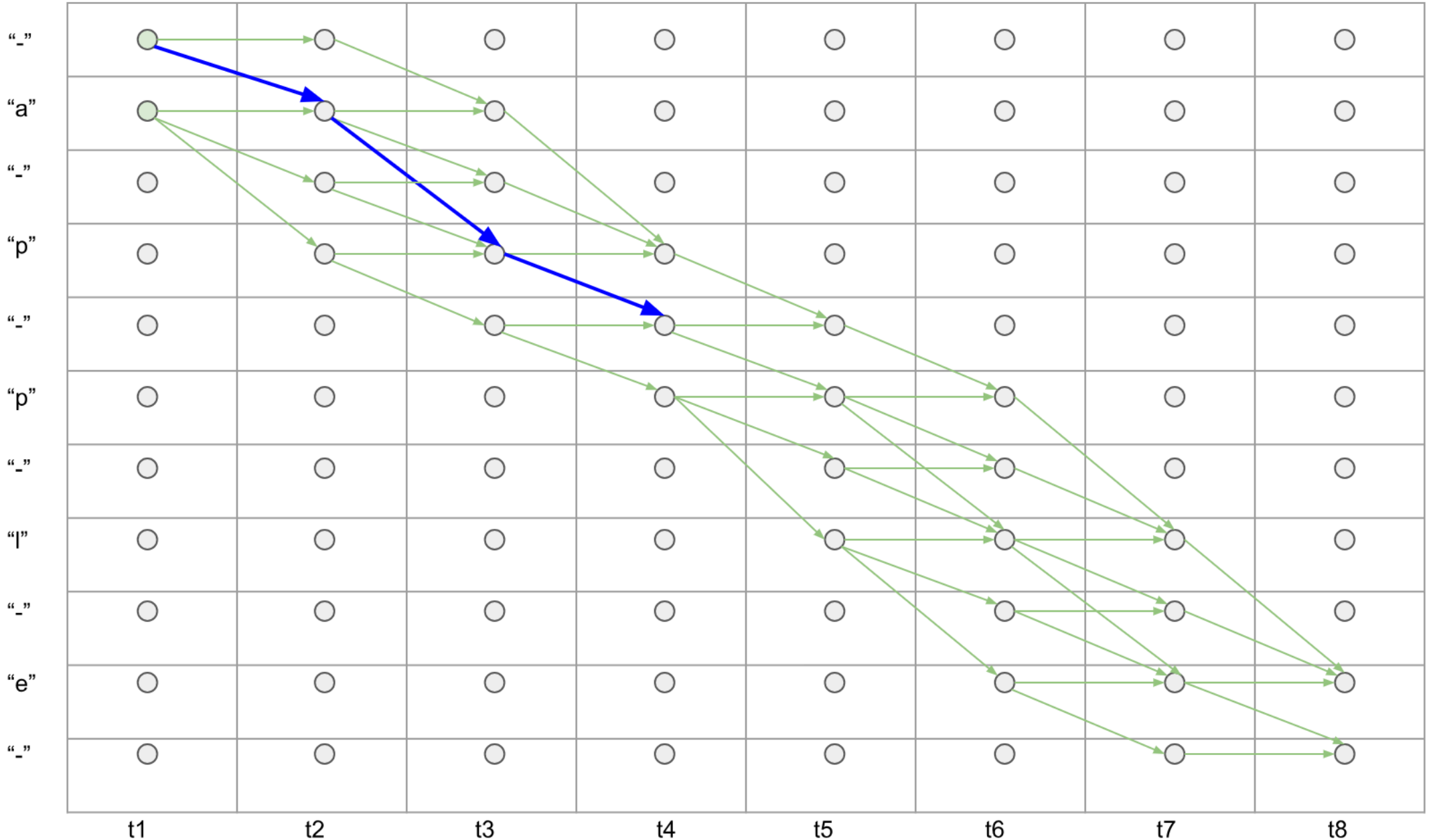
Initialization: paths can start only with this symbols.

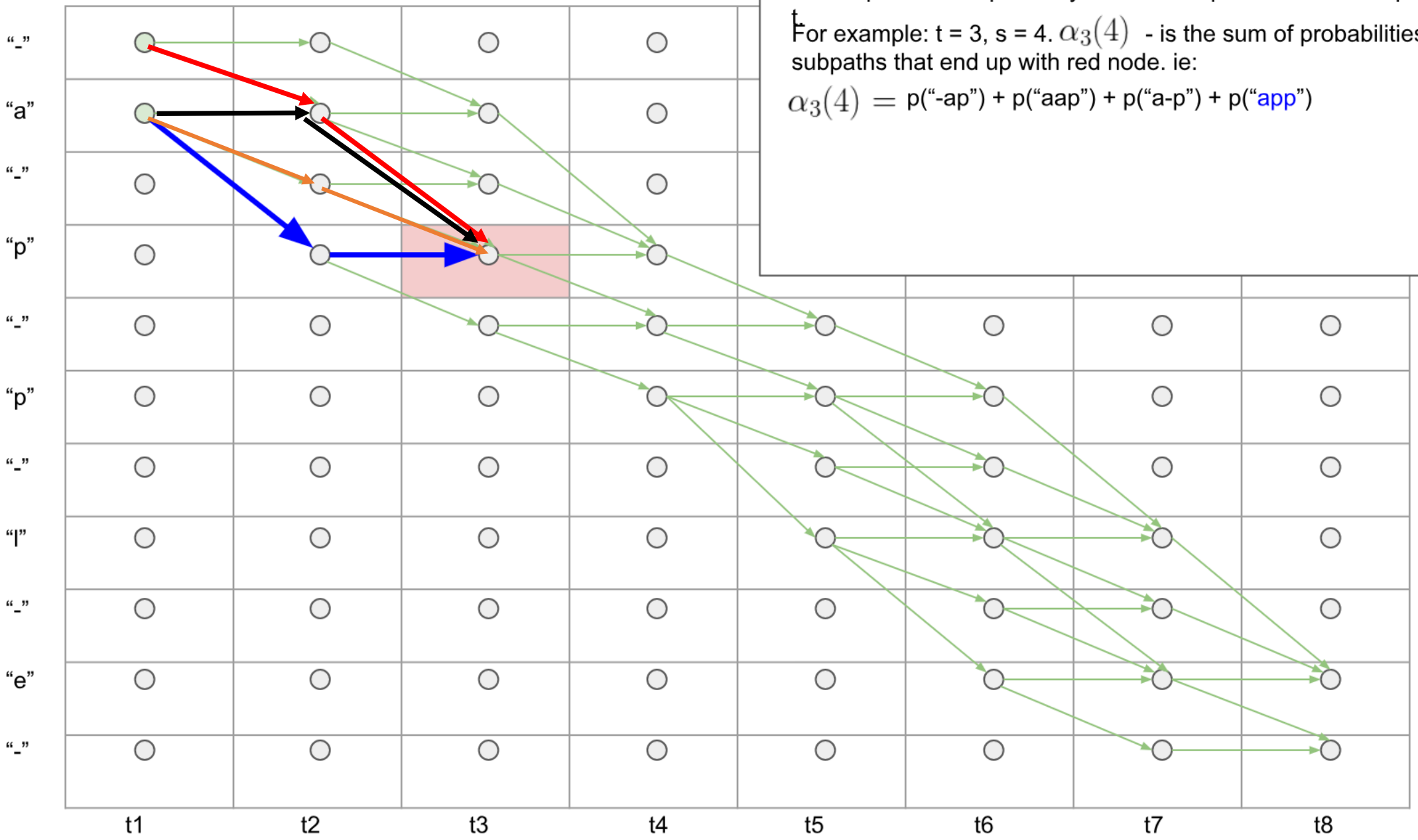
	t1	t2	t3	t4	t5	t6	t7	t8
“-”	○	○	○	○	○	○	○	○
“a”	○	○	○	○	○	○	○	○
“-”	○	○	○	○	○	○	○	○
“p”	○	○	○	○	○	○	○	○
“-”	○	○	○	○	○	○	○	○
“p”	○	○	○	○	○	○	○	○
“-”	○	○	○	○	○	○	○	○
“l”	○	○	○	○	○	○	○	○
“-”	○	○	○	○	○	○	○	○
“e”	○	○	○	○	○	○	○	○
“-”	○	○	○	○	○	○	○	○

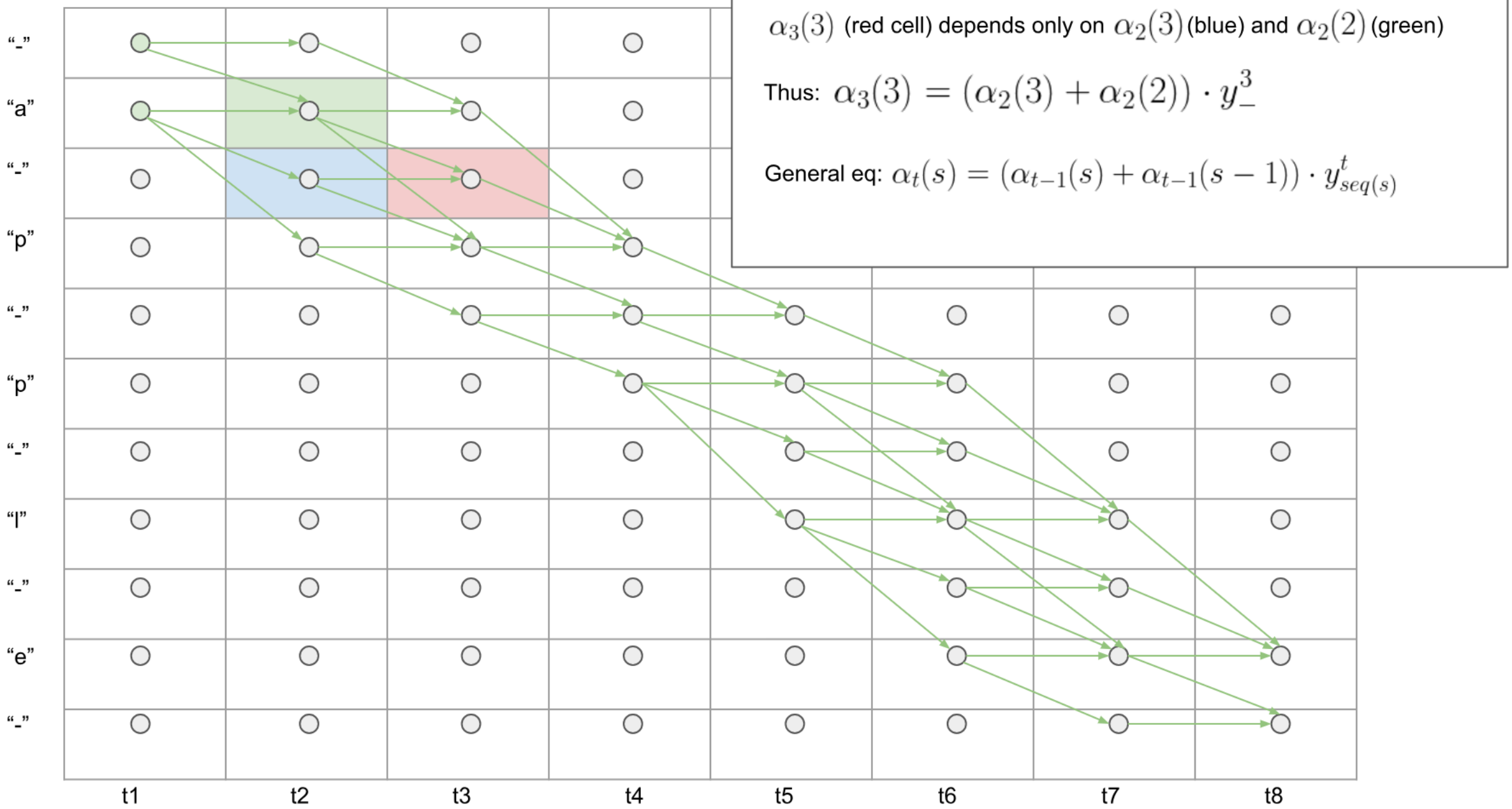
Interesting note: all valid paths should end up with this nodes.

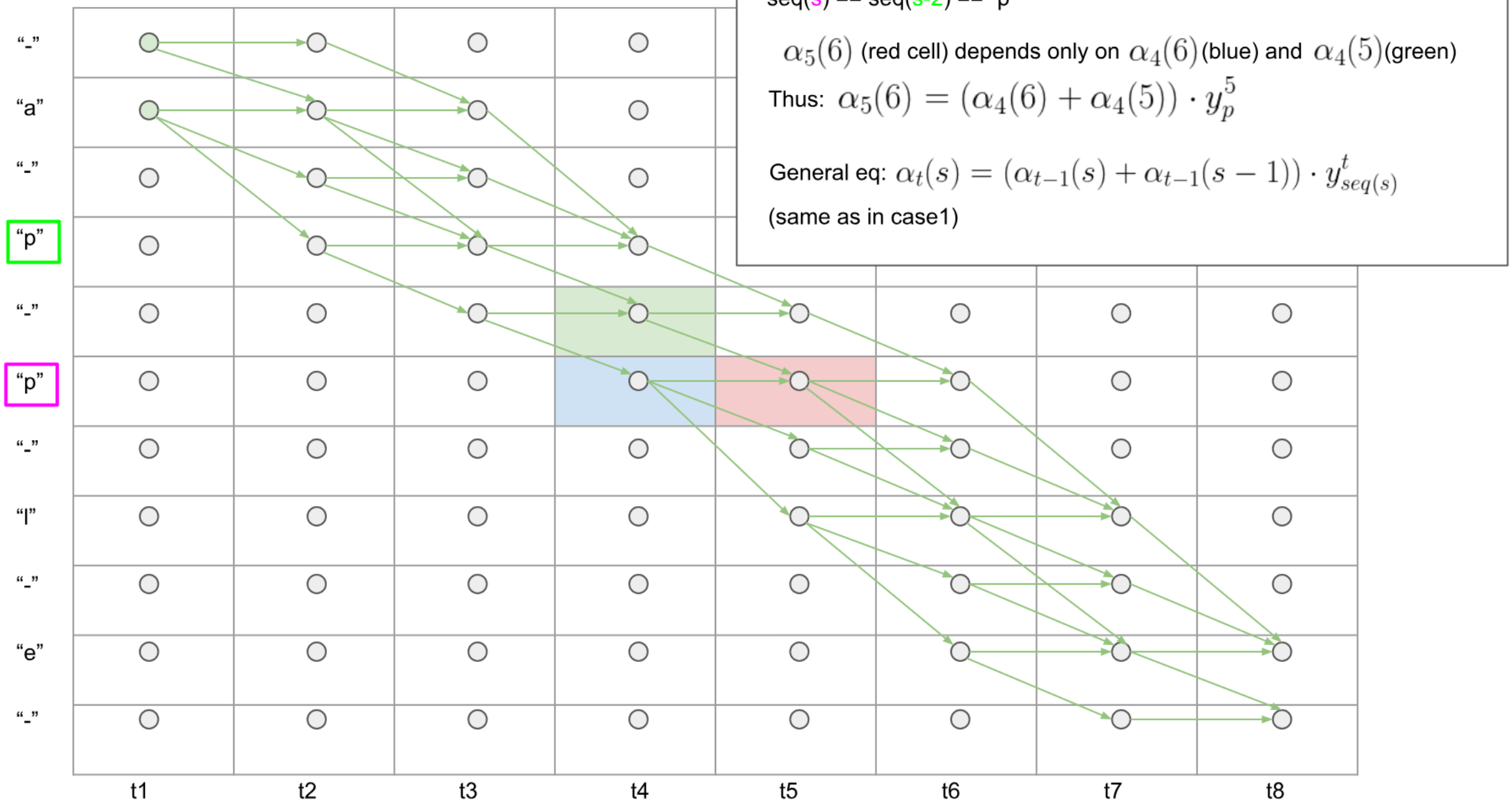


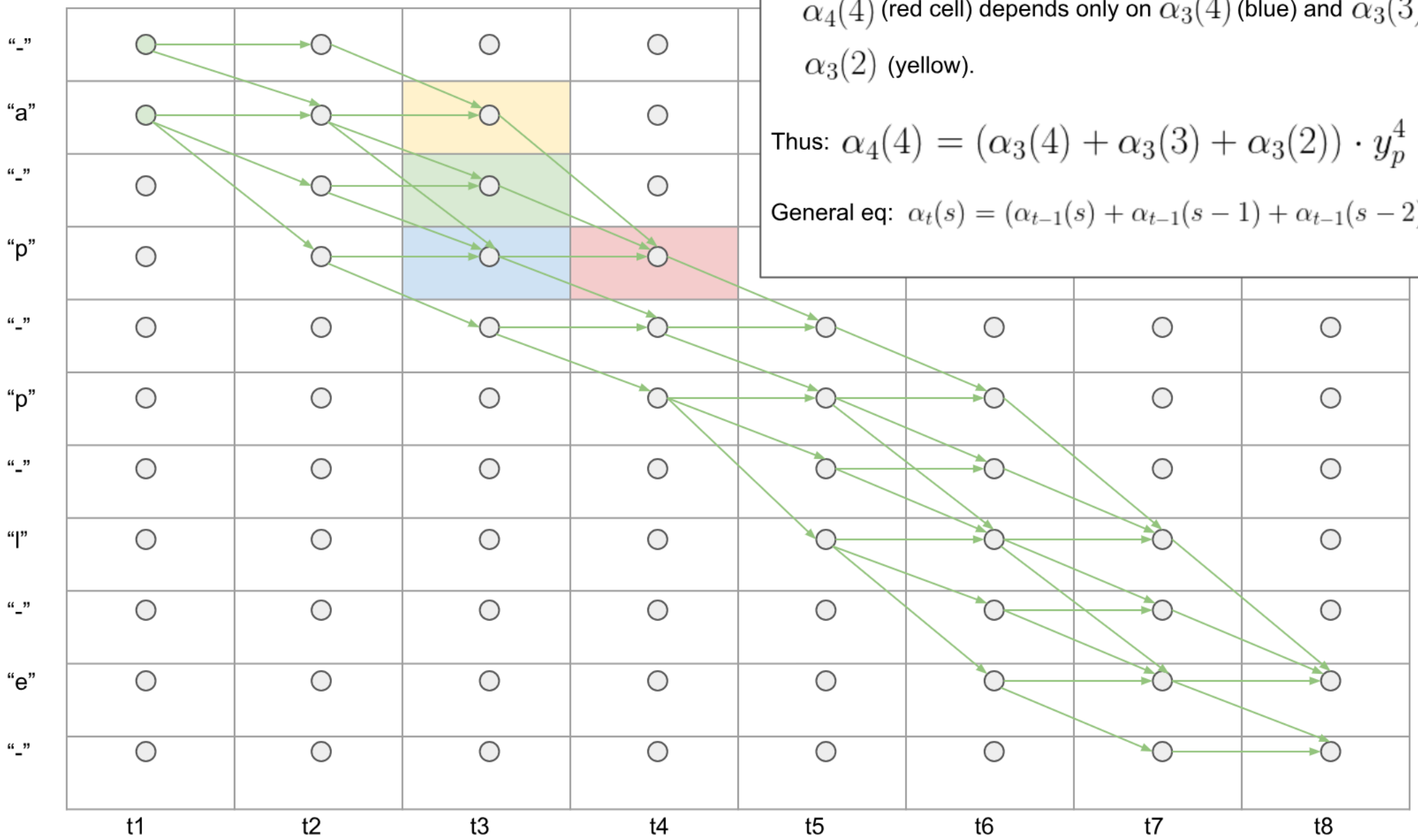
Let's consider subpath "-ap-" (blue color). Let me remind you that the probability of this subpath is simply the multiplication of corresponding network output probabilities - $p(\text{“}-ap-\text{”}) = y^1_- \cdot y^2_a \cdot y^3_p \cdot y^4_-$




Let me define variable $\alpha_t(s)$ - the total probability of all subpaths, whose prefix end up with symbol at s-th position in the sequence at time t.
For example: t = 3, s = 4. $\alpha_3(4)$ - is the sum of probabilities of all subpaths that end up with red node. ie:
$\alpha_3(4)$ = p("-ap") + p("aap") + p("a-p") + p("app")
"-"
"a"
"-"
"p"
"-"
"p"
"-"
"l"
"-"
"e"
"-"
t1
t2
t3
t4
t5
t6
t7
t8


Case1. s-th symbol is blank. Example: s=3, 3-rd symbol is “-”
$\alpha_3(3)$ (red cell) depends only on $\alpha_2(3)$ (blue) and $\alpha_2(2)$ (green)
Thus: $\alpha_3(3) = (\alpha_2(3) + \alpha_2(2)) \cdot y^3_-$
General eq: $\alpha_t(s) = (\alpha_{t-1}(s) + \alpha_{t-1}(s-1)) \cdot y^t_{seq(s)}$
“-”
“a”
“-”
“p”
“-”
“p”
“-”
“l”
“-”
“e”
“-”
t1
t2
t3
t4
t5
t6
t7
t8


Case2. s-th symbol is equal to (s-2)-th symbol. Example: s=6,t=5.
seq(s) == seq(s-2) == "p"
$\alpha_5(6)$ (red cell) depends only on $\alpha_4(6)$ (blue) and $\alpha_4(5)$ (green)
Thus: $\alpha_5(6) = (\alpha_4(6) + \alpha_4(5)) \cdot y_p^5$
General eq: $\alpha_t(s) = (\alpha_{t-1}(s) + \alpha_{t-1}(s-1)) \cdot y_{seq(s)}^t$
(same as in case1)
"-"
"a"
"-"
"p"
"-"
"p"
"-"
"l"
"-"
"e"
"-"
t1
t2
t3
t4
t5
t6
t7
t8

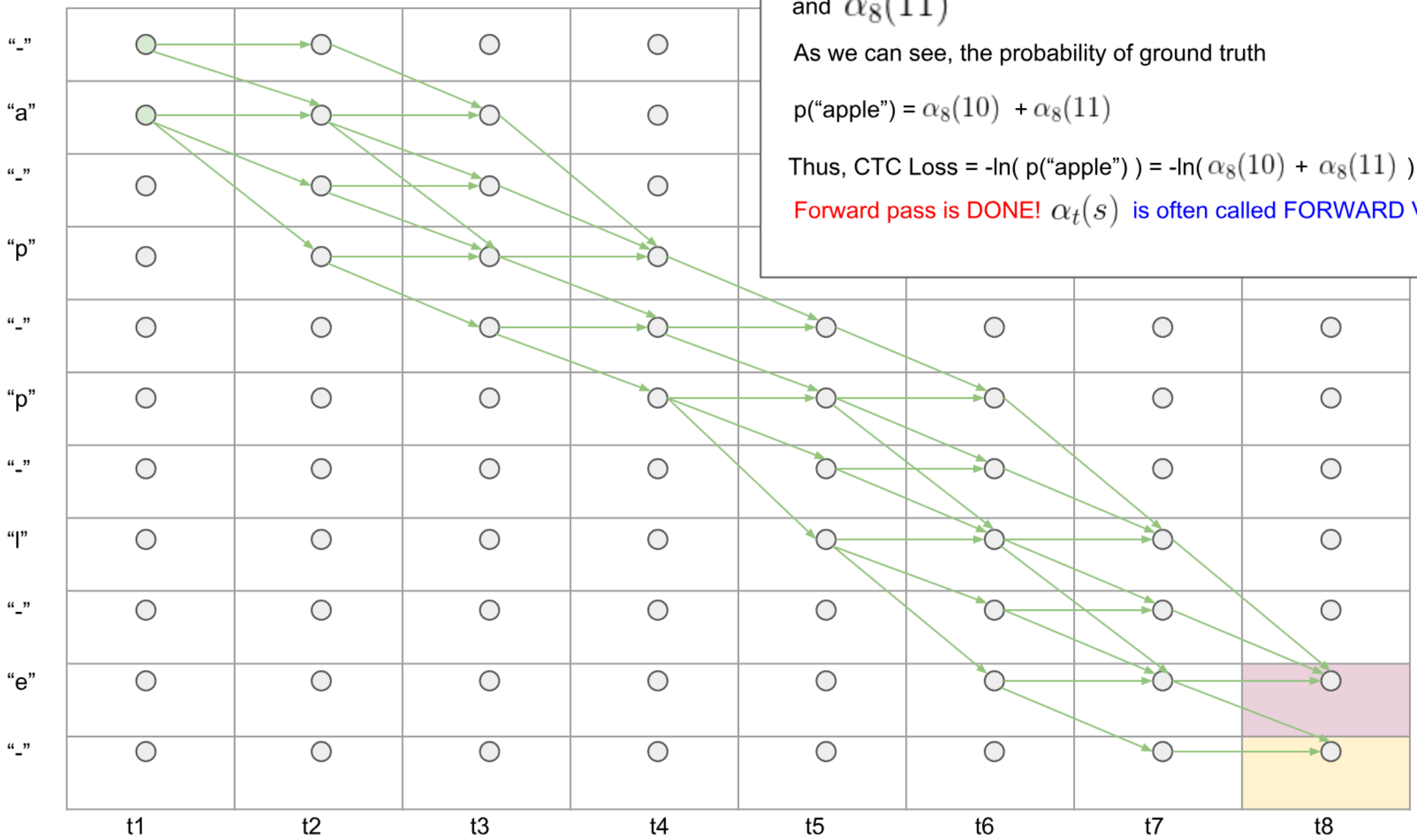


Case3. Otherwise. Example: s=4,t=4.

$\alpha_4(4)$ (red cell) depends only on $\alpha_3(4)$ (blue) and $\alpha_3(3)$ (green) and $\alpha_3(2)$ (yellow).

Thus: $\alpha_4(4) = (\alpha_3(4) + \alpha_3(3) + \alpha_3(2)) \cdot y_p^4$

General eq: $\alpha_t(s) = (\alpha_{t-1}(s) + \alpha_{t-1}(s-1) + \alpha_{t-1}(s-2)) \cdot y_{seq(s)}^t$



With this dynamic programming algorithm we can calculate $\alpha_8(10)$ and $\alpha_8(11)$

As we can see, the probability of ground truth

p("apple") = $\alpha_8(10)$ + $\alpha_8(11)$

Thus, CTC Loss = -ln(p("apple")) = -ln($\alpha_8(10)$ + $\alpha_8(11)$)

Forward pass is DONE! $\alpha_t(s)$ is often called FORWARD VARIABLE.

总结：前向递推公式

This gives us the following rules for initialisation

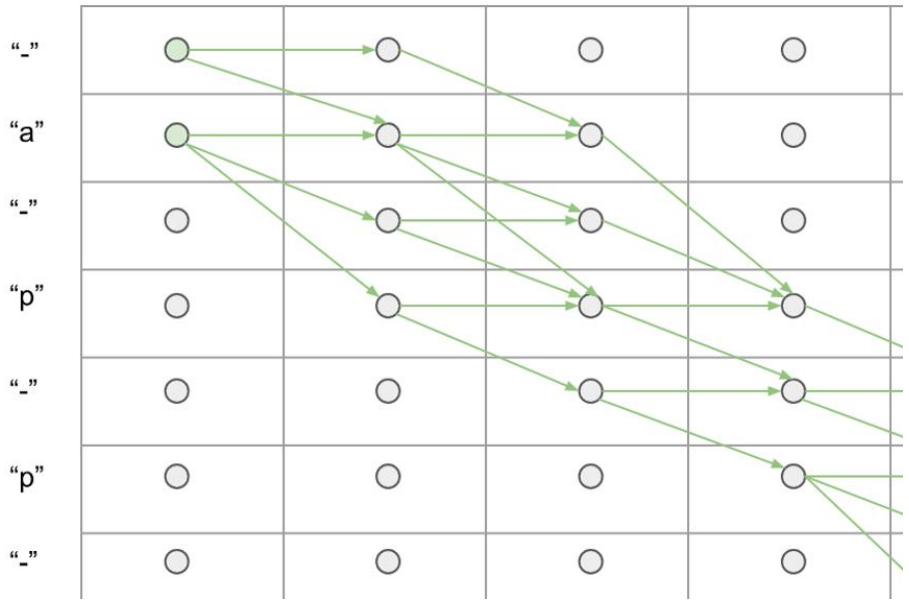
$$\alpha_1(1) = y_b^1$$
$$\alpha_1(2) = y_{\mathrm{l}_1}^1$$
$$\alpha_1(s) = 0, \ \forall s > 2$$

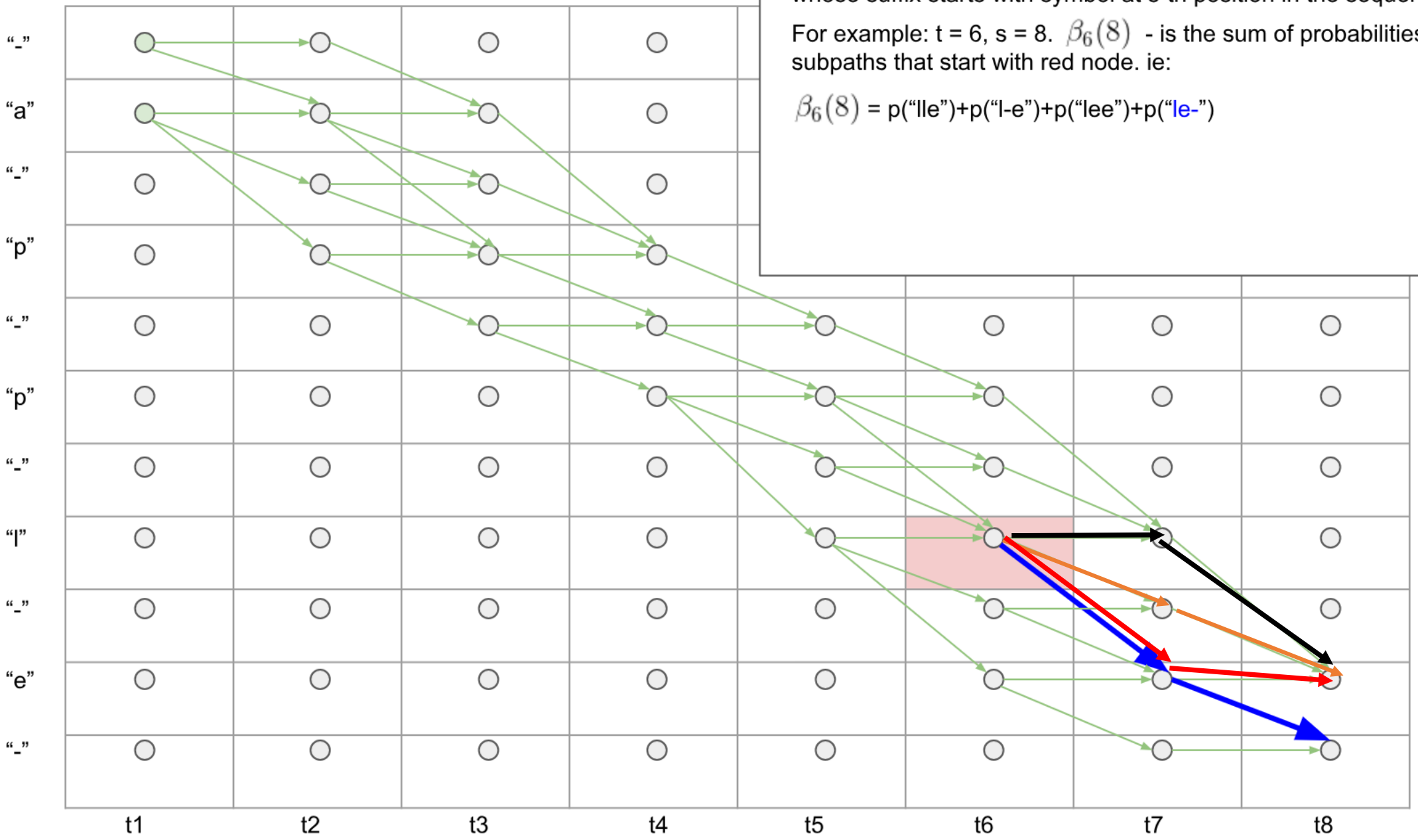
and recursion

$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y_{\mathrm{l}'_s}^t & \text{if } \mathrm{l}'_s = b \text{ or } \mathrm{l}'_{s-2} = \mathrm{l}'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y_{\mathrm{l}'_s}^t & \text{otherwise} \end{cases} \tag{6}$$

where

$$\bar{\alpha}_t(s) \stackrel{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1). \tag{7}$$




Let me define variable $\beta_t(s)$ - the total probability of all subpaths, whose suffix starts with symbol at s-th position in the sequence at time t.
For example: t = 6, s = 8. $\beta_6(8)$ - is the sum of probabilities of all subpaths that start with red node. ie:
$\beta_6(8)$ = p("lle")+p("l-e")+p("lee")+p("le-")
"-"
"a"
"-"
"p"
"-"
"p"
"-"
"l"
"-"
"e"
"-"
t1
t2
t3
t4
t5
t6
t7
t8

总结：后向递推公式

Similarly, the backward variables $\beta_t(s)$ are defined as the total probability of $\mathbf{l}_{s:|\mathbf{l}|}$ at time t.

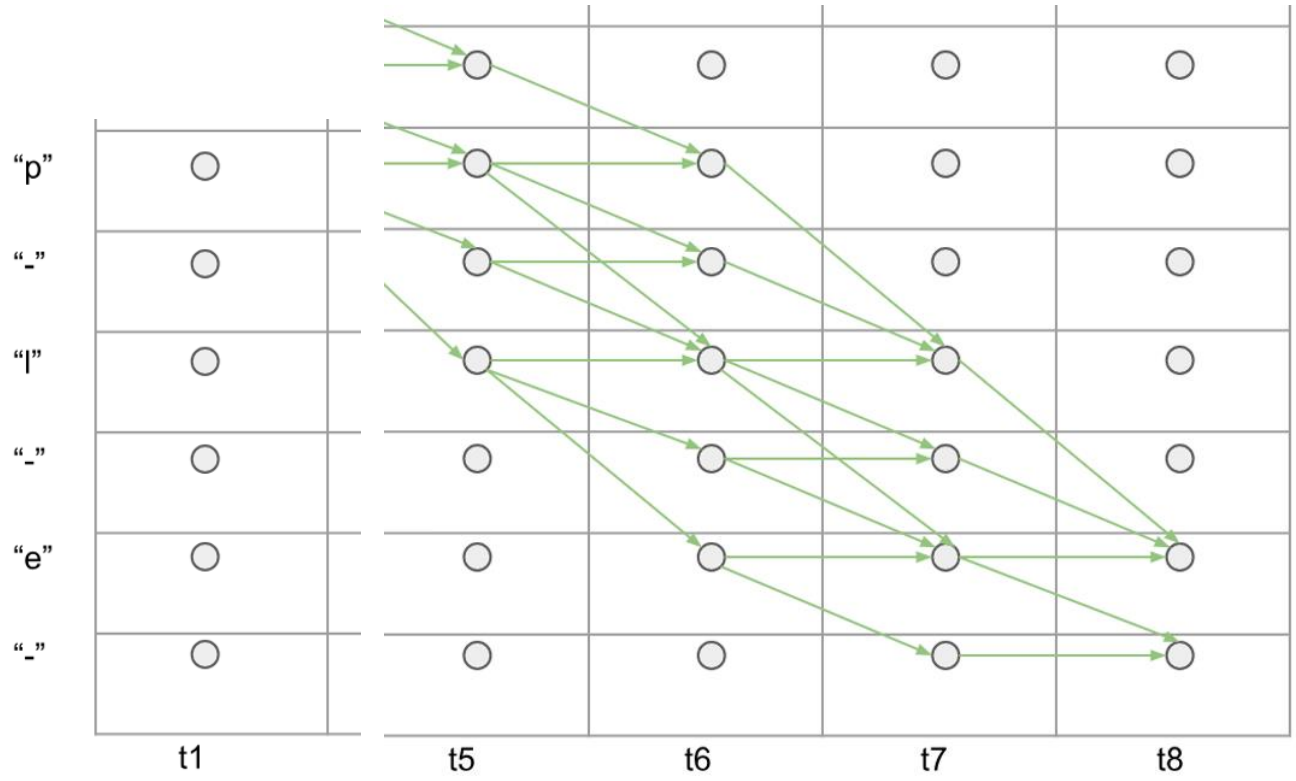
$$\beta_t(s) \overset{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{t:T}) = \mathbf{l}_{s:|\mathbf{l}|}}} \prod_{t'=t}^{T} y^{t'}_{\pi_{t'}} \tag{9}$$

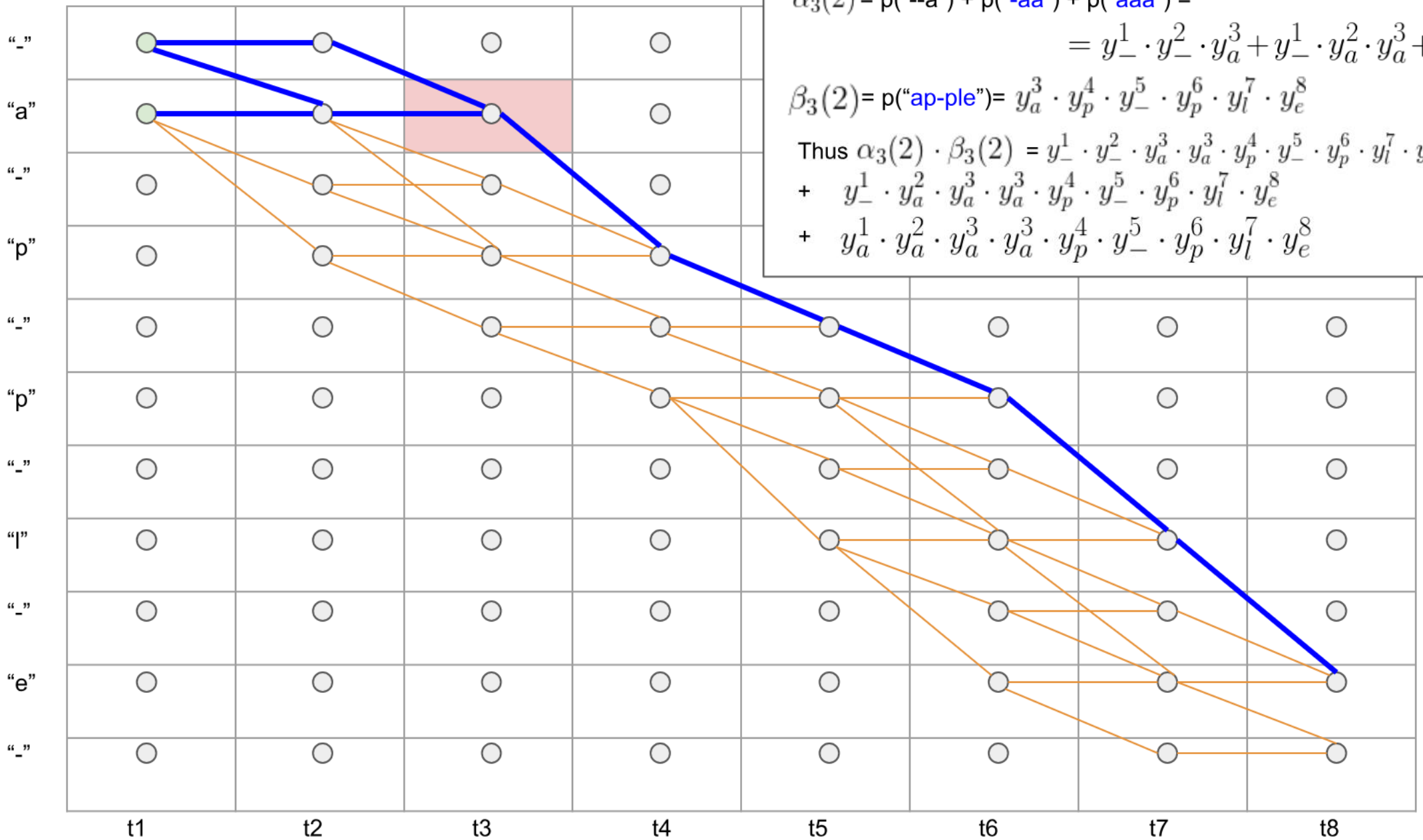
$$\begin{aligned} \beta_T(|\mathbf{l}'|) &= y^T_b \\ \beta_T(|\mathbf{l}'|-1) &= y^T_{\mathbf{l}_{|\mathbf{l}|}} \\ \beta_T(s) &= 0, \ \forall s < |\mathbf{l}'|-1 \end{aligned}$$

$$\beta_t(s) = \begin{cases} \bar{\beta}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s+2} = \mathbf{l}'_s \\ \big(\bar{\beta}_t(s) + \beta_{t+1}(s+2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases} \tag{10}$$

where

$$\bar{\beta}_t(s) \overset{\text{def}}{=} \beta_{t+1}(s) + \beta_{t+1}(s+1). \tag{11}$$





Let's consider t=3,s=2.

$\alpha_3(2)$ = p("--a") + p("-aa") + p("aaa") =

$$= y_-^1 \cdot y_-^2 \cdot y_a^3 + y_-^1 \cdot y_a^2 \cdot y_a^3 + y_a^1 \cdot y_a^2 \cdot y_a^3$$

$\beta_3(2)$= p("ap-ple")= $y_a^3 \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$

Thus $\alpha_3(2) \cdot \beta_3(2) = y_-^1 \cdot y_-^2 \cdot y_a^3 \cdot y_a^3 \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$ +

+ $y_-^1 \cdot y_a^2 \cdot y_a^3 \cdot y_a^3 \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$

+ $y_a^1 \cdot y_a^2 \cdot y_a^3 \cdot y_a^3 \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$

Thus $\alpha_3(2) \cdot \beta_3(2) = y_-^1 \cdot y_-^2 \cdot y_a^3 \cdot \boxed{y_a^3} \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$ +

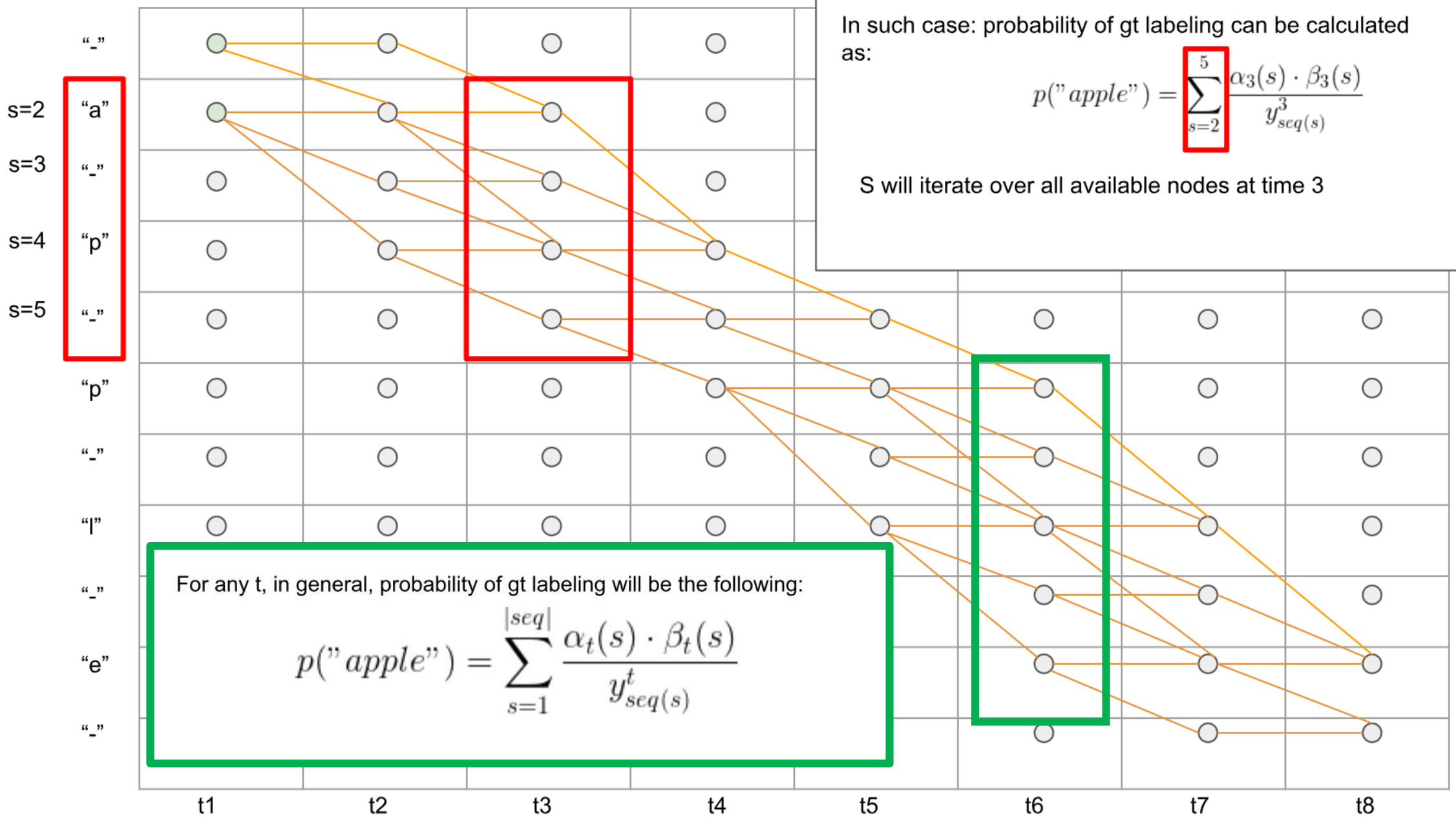
+ $y_-^1 \cdot y_a^2 \cdot y_a^3 \cdot \boxed{y_a^3} \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$

+ $y_a^1 \cdot y_a^2 \cdot y_a^3 \cdot \boxed{y_a^3} \cdot y_p^4 \cdot y_-^5 \cdot y_p^6 \cdot y_l^7 \cdot y_e^8$ =

= (p("--aap-ple") + p("-aap-ple") + p("aaap-ple")) * y_a^3

$\alpha_3(2) \cdot \beta_3(2)$ = (p("--aap-ple") + p("-aap-ple") + p("aaap-ple")) * y_a^3

$\frac{\alpha_3(2) \cdot \beta_3(2)}{y_a^3}$ = sum of probabilities of all paths that go through second sequence symbol (symbol "a") at time 3



Let’s commit: t=3

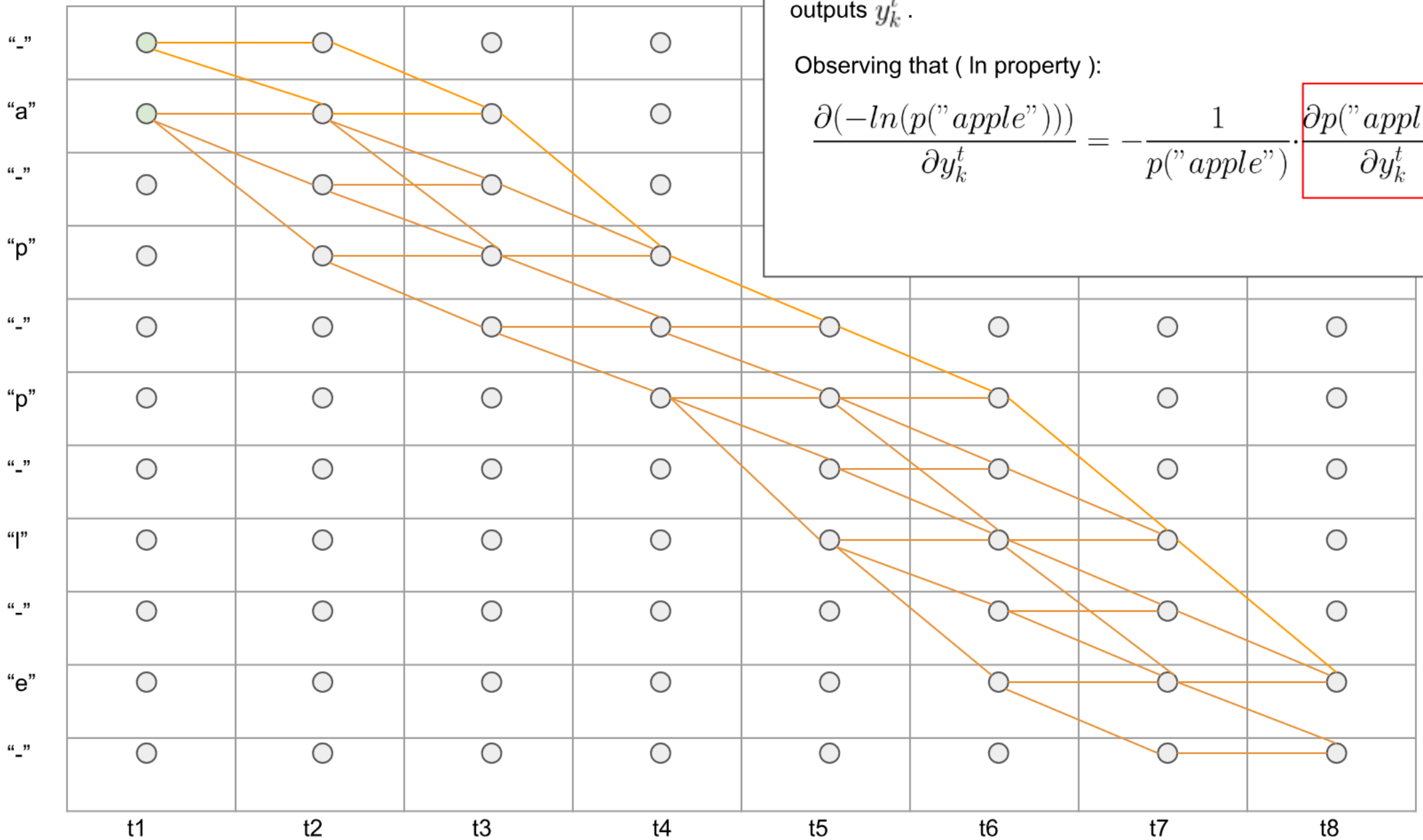
In such case: probability of gt labeling can be calculated as:

$$p(\text{"apple"}) = \sum_{s=2}^{5} \frac{\alpha_3(s) \cdot \beta_3(s)}{y^3_{seq(s)}}$$

S will iterate over all available nodes at time 3

For any t, in general, probability of gt labeling will be the following:

$$p(\text{"apple"}) = \sum_{s=1}^{|seq|} \frac{\alpha_t(s) \cdot \beta_t(s)}{y^t_{seq(s)}}$$



To do backprop, we should differentiate loss with respect to all network outputs y_k^t .

Observing that (ln property):

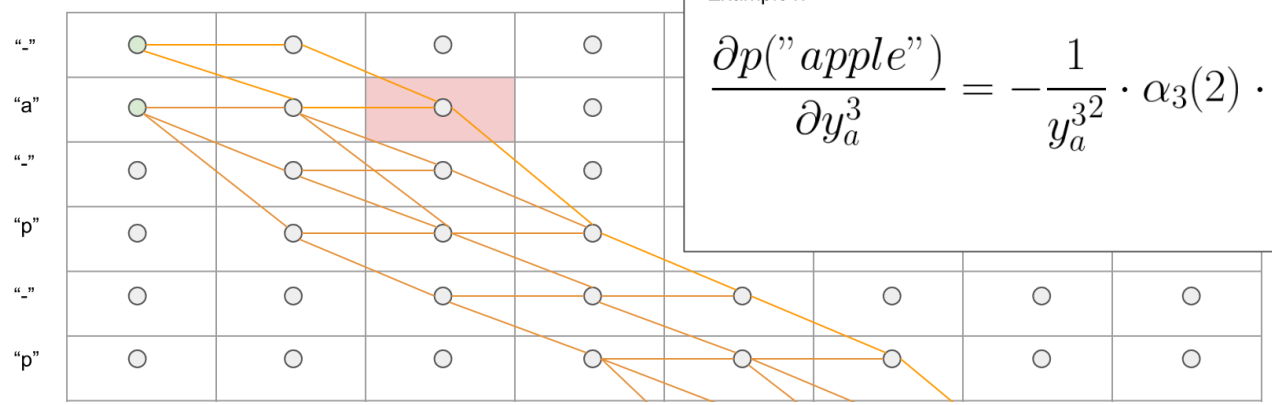
$$\frac{\partial(-ln(p(”apple”)))}{\partial y_k^t} = -\frac{1}{p(”apple”)} \cdot \frac{\partial p(”apple”)}{\partial y_k^t}$$

$$\frac{\partial p(\text{"apple"})}{\partial y_k^t}$$

To differentiate this with respect to y_k^t we need only consider those paths going through symbol k at time t.

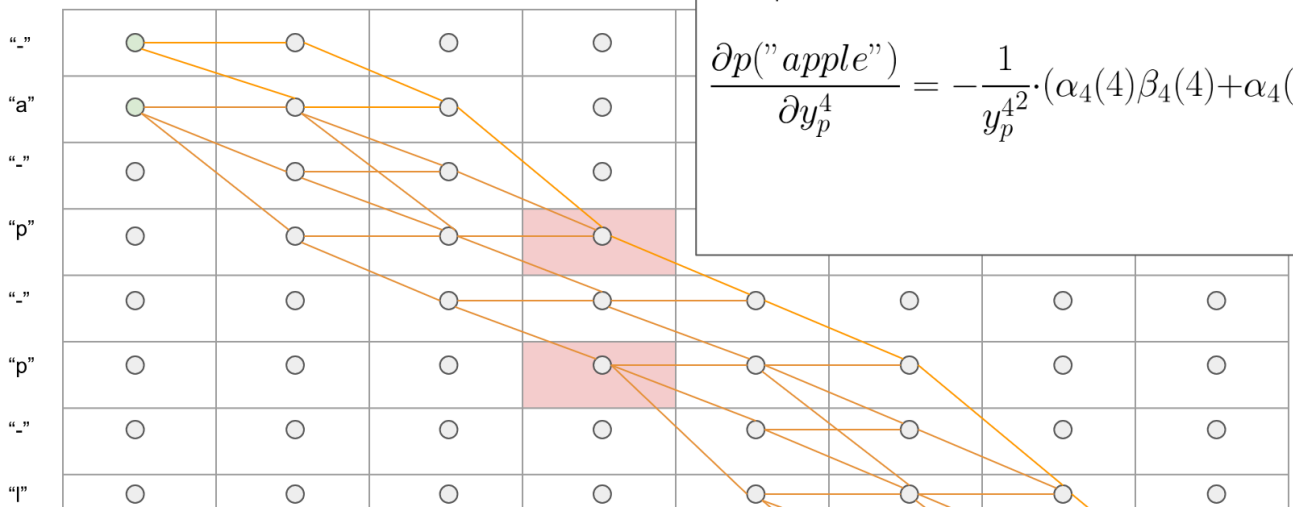
$$\frac{\partial p(\text{"apple"})}{\partial y_k^t} = -\frac{1}{{y_k^t}^2} \cdot \sum_{s:seq(s)=k} \alpha_t(s) \cdot \beta_t(s)$$

此处应为"+"



Example1:

$$\frac{\partial p(\text{"apple"})}{\partial y_a^3} = -\frac{1}{y_a^{3^2}} \cdot \alpha_3(2) \cdot \beta_3(2)$$



Example2:

$$\frac{\partial p(\text{"apple"})}{\partial y_p^4} = -\frac{1}{y_p^{4^2}} \cdot (\alpha_4(4)\beta_4(4) + \alpha_4(6)\beta_4(6))$$

Conclusion

- Dynamic programming
- Matrix α (forward variables) is used to compute loss
- Matrix β (backward variables) is used to compute gradients

CTC解码

Best path decoding

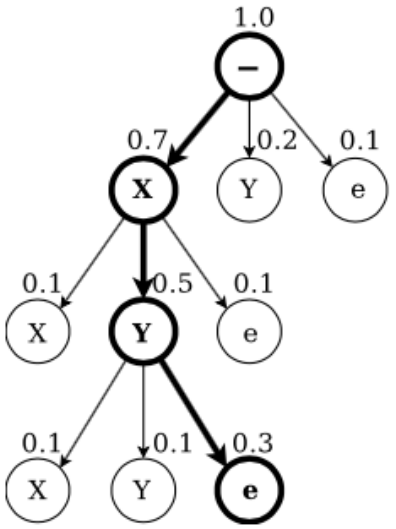
按照上面思路，需要找到序列 I 的所有路径的概率，一种简化的方式是：找到路径中概率最大的，然后其对应的序列 I 就是最优序列，这个方法被称为「Best path decoding」。

这个方法相当于是假设，最优序列的最优路径也是全局最优的（最优表示概率最大），形式化表示为：

$$h(\mathbf{x}) \approx \mathcal{B}(\pi^*)$$
$$\text{where } \pi^* = \arg\max_{\pi \in N^t} p(\pi|\mathbf{x}).$$

prefix search decoding

接着介绍第二种方法，「prefix search decoding」是一种剪枝的算法，剪枝主要在两个方面，一是同路径不重复计算，二是不可能状态不再搜索，下图中第一层的Y不搜索就是因为同层的X和下层的Y概率都比他高。



References

[1] Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks

[2] https://blog.csdn.net/luodongri/article/details/77005948

[3] https://distill.pub/2017/ctc/

[4]https://docs.google.com/presentation/d/12gYcPft9_4cxk2AD6Z6ZlJNa3wvZCW1ms31nhq51vMk/pub?start=false&loop=false&delayms=3000&slide=id.g24e9f0de4f_0_21796

[5] https://www.youtube.com/watch?v=eYIL4TMAeRI&t=549s

[6] https://www.youtube.com/watch?v=c86gfVGcvh4