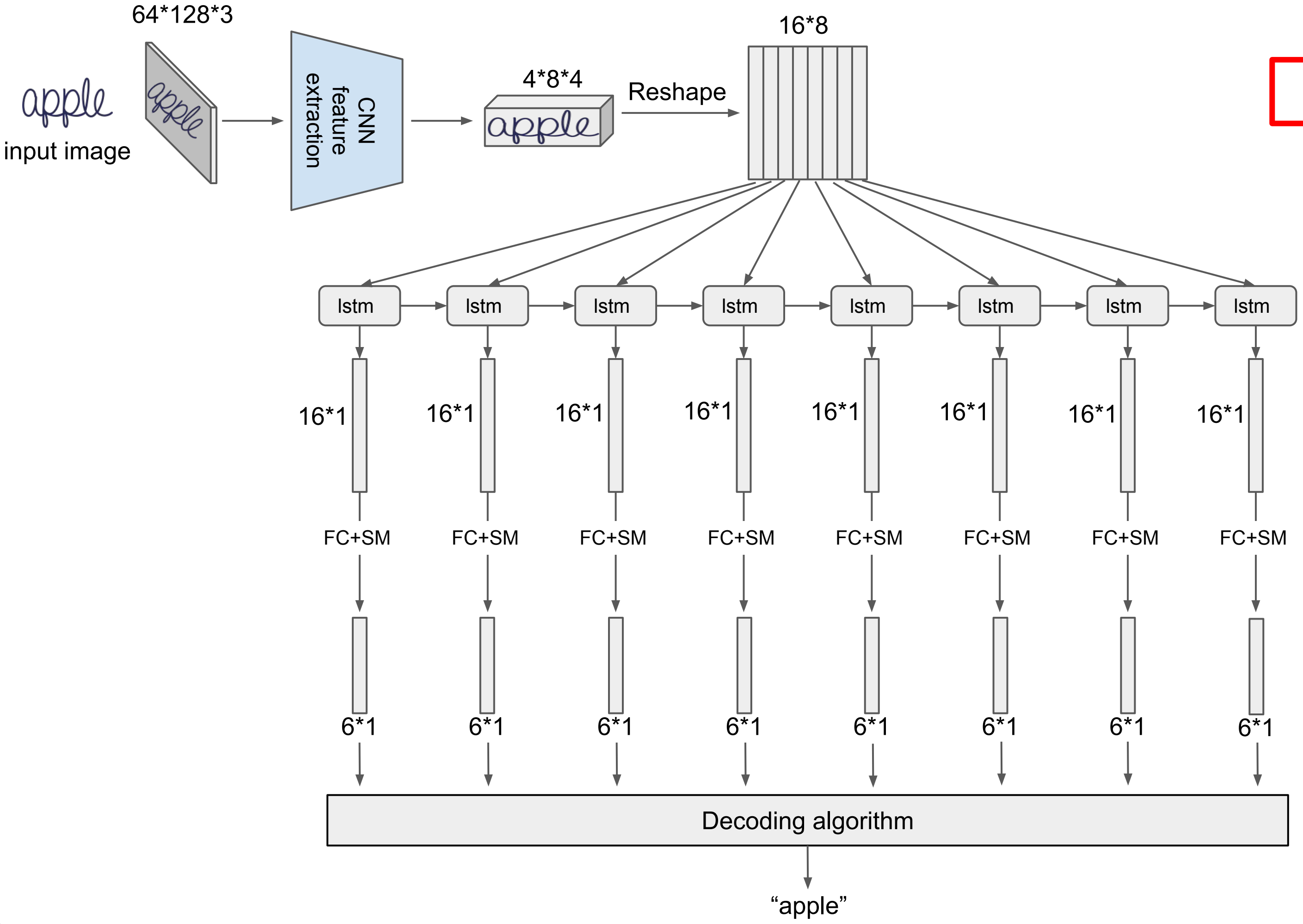
Review from deepsystems.ai

# How to build end-to-end recognition system (Part 2): CTC Loss (Alex Graves).

“apple”

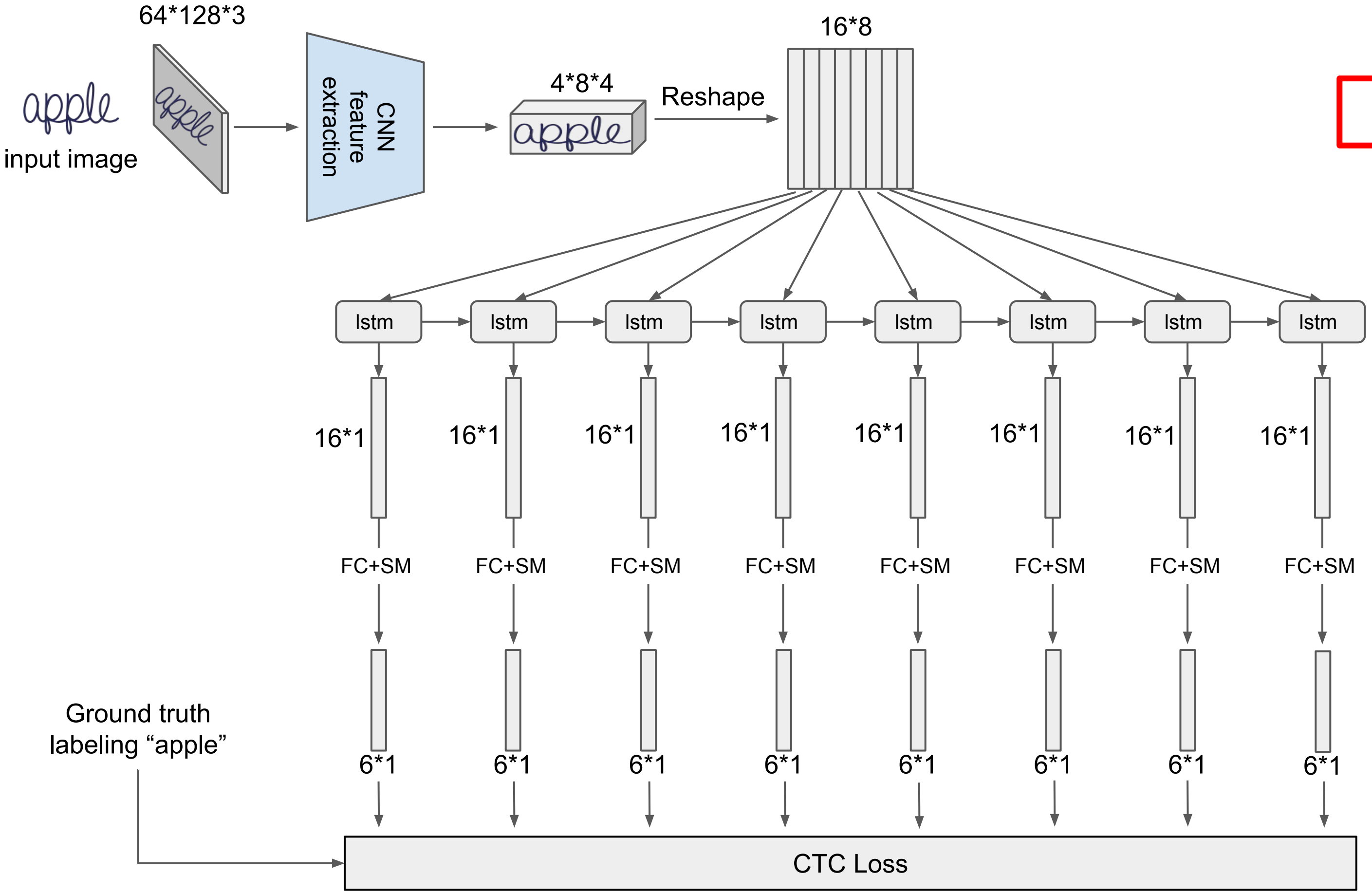


Link to previous lecture

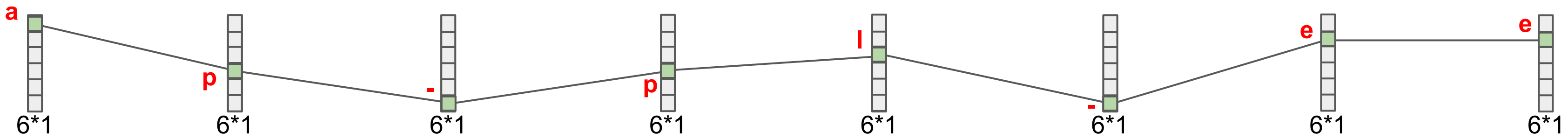
deepsystems.ai

# Image OCR: model architecture

## Training: CTC Loss

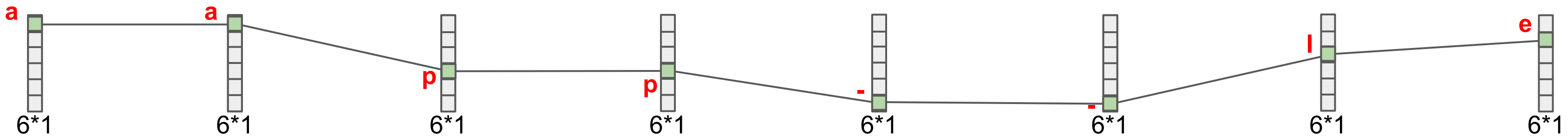


Link to CTC Loss paper



Path1: “ap-pl-ee” $\xrightarrow{\text{B(“ap-pl-ee”)}}$ Labeling: “apple”

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



Path1: “ap-pl-ee” $\xrightarrow{B(\text{“ap-pl-ee”})}$ 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” $\xrightarrow{B(\text{“aapp--le”})}$ Labeling: “aple”

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

6*1
6*1
6*1
6*1
6*1
6*1
6*1
6*1

Path1: “ap-pl-ee” $\xrightarrow{\text{B(“ap-pl-ee”)}}$ Labeling: “apple”

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

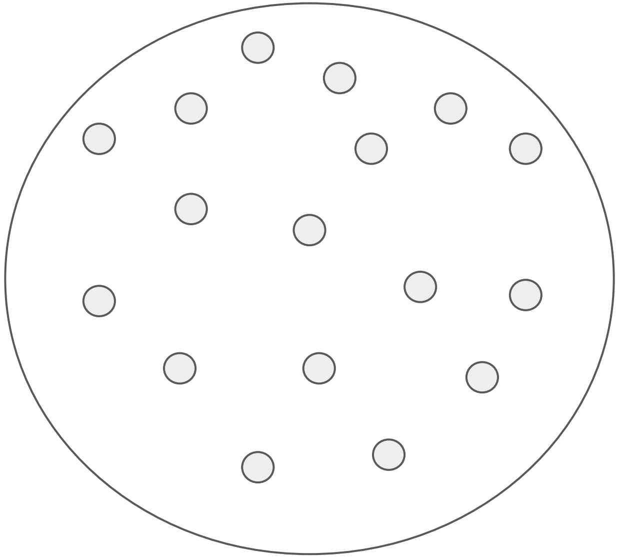
Path2: “aapp--le” $\xrightarrow{\text{B(“aapp--le”)}}$ Labeling: “aple”

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

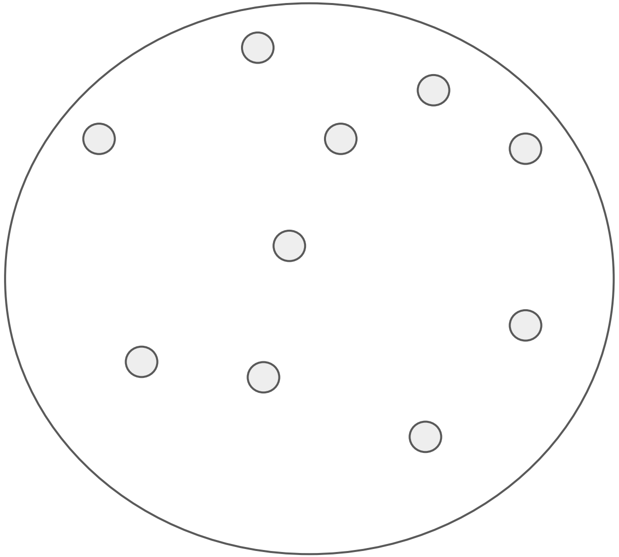
Path3: “aap--lzz” $\xrightarrow{\text{B(“aap--lzz”)}}$ Labeling: “aplz”

... ... ...

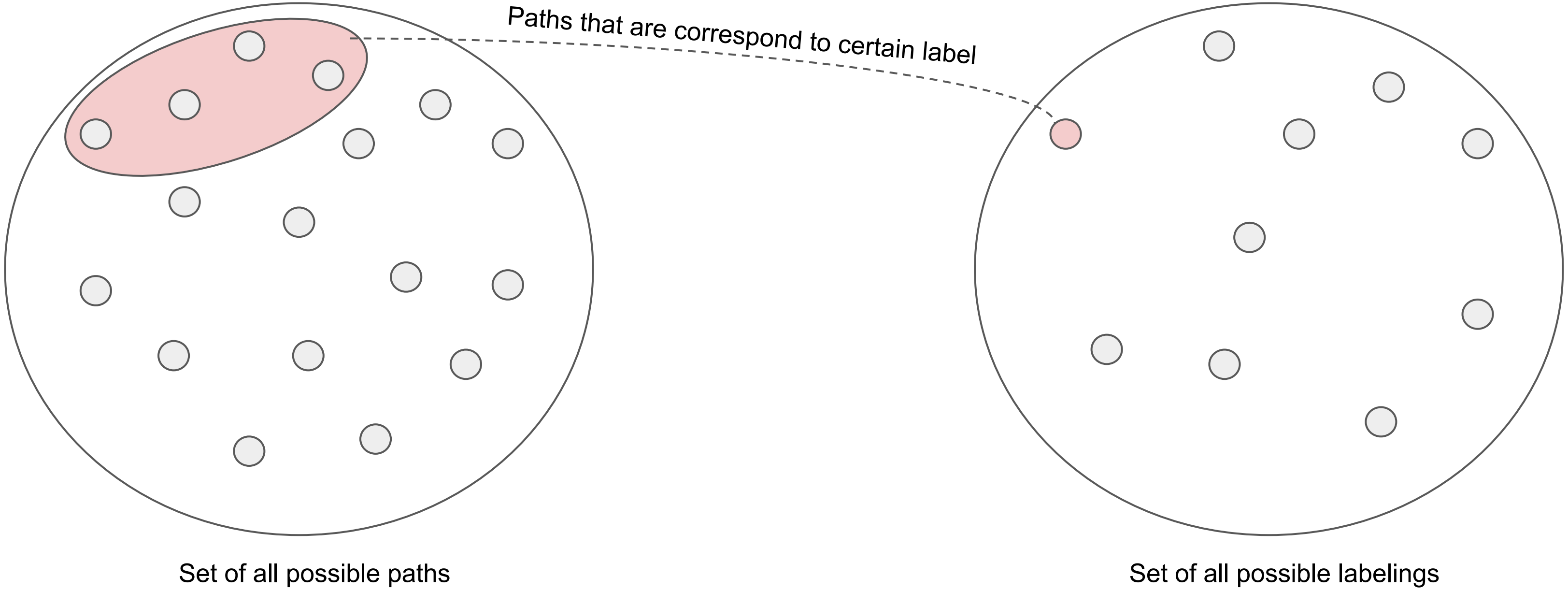
PathN: “--z--eee” $\xrightarrow{\text{B(“--z--eee”)}}$ Labeling: “ze”

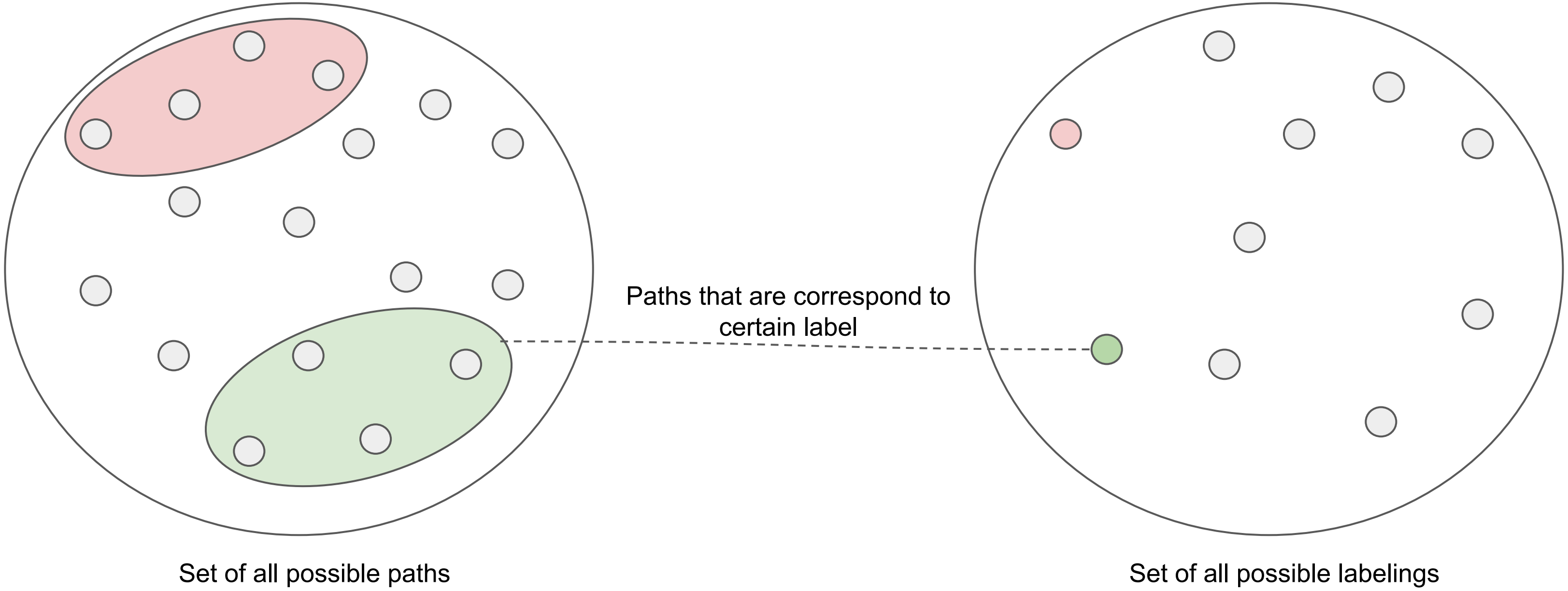


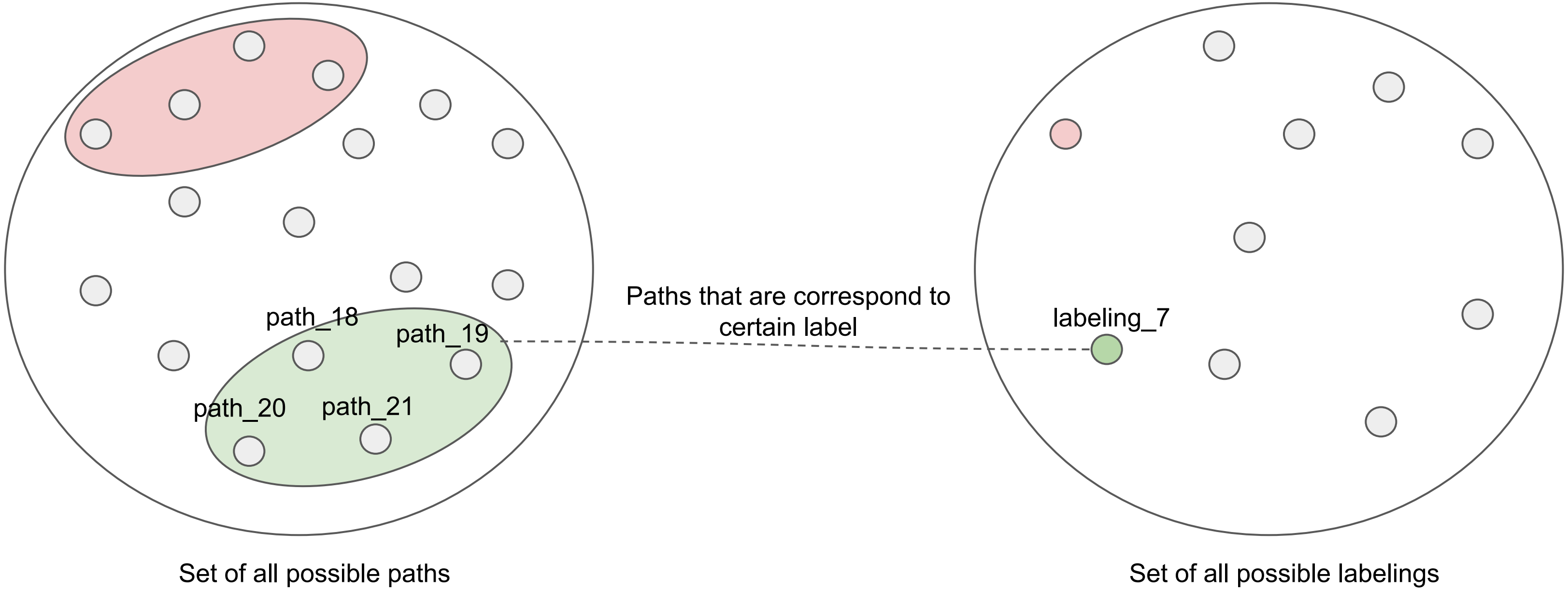
Set of all possible paths

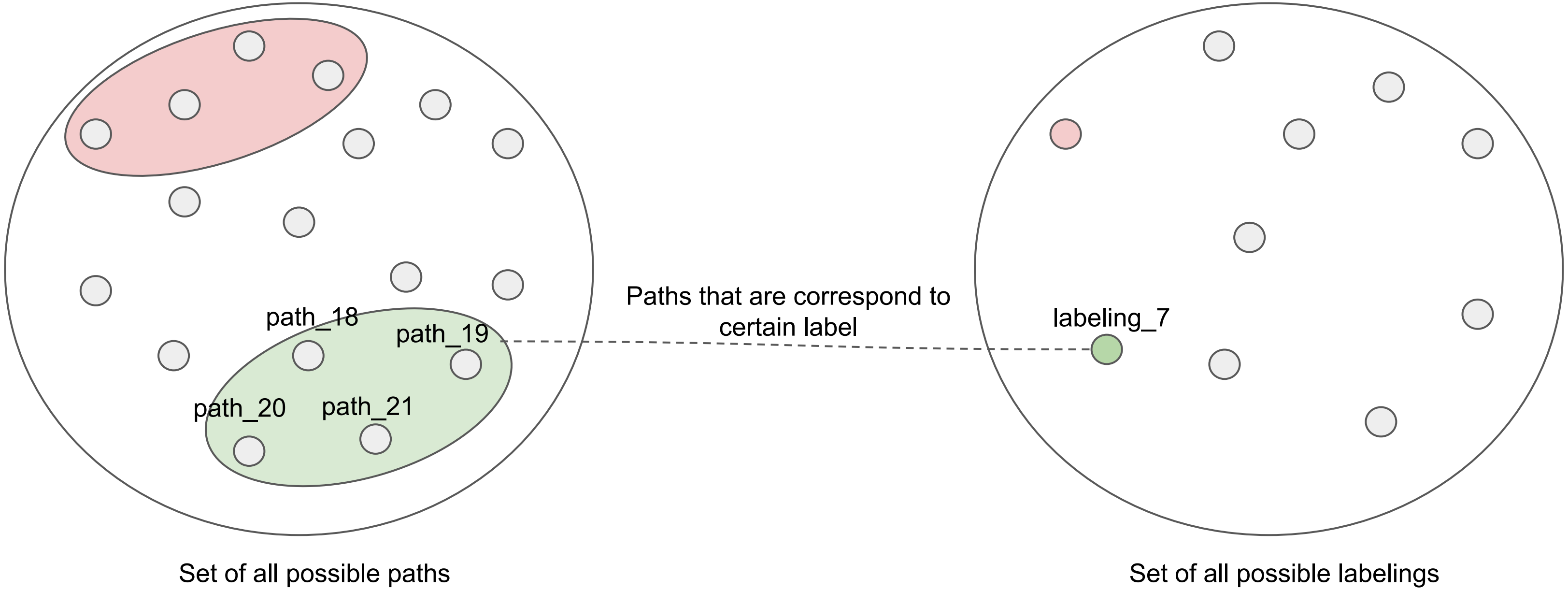


Set of all possible labelings

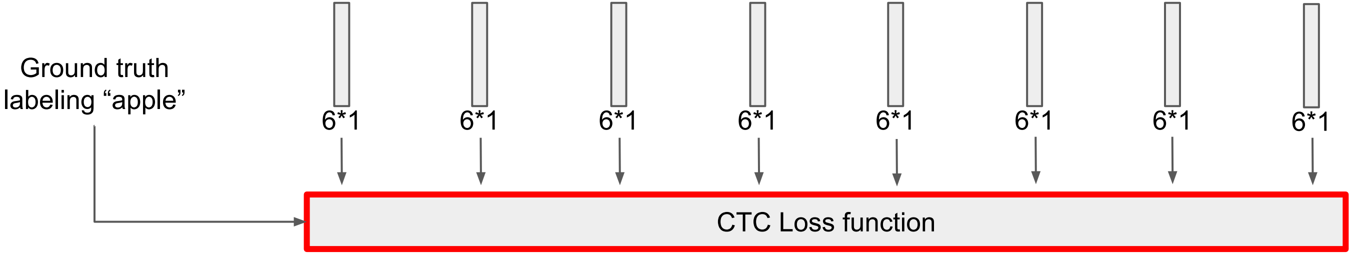

Paths that are correspond to certain label
Set of all possible paths
Set of all possible labelings

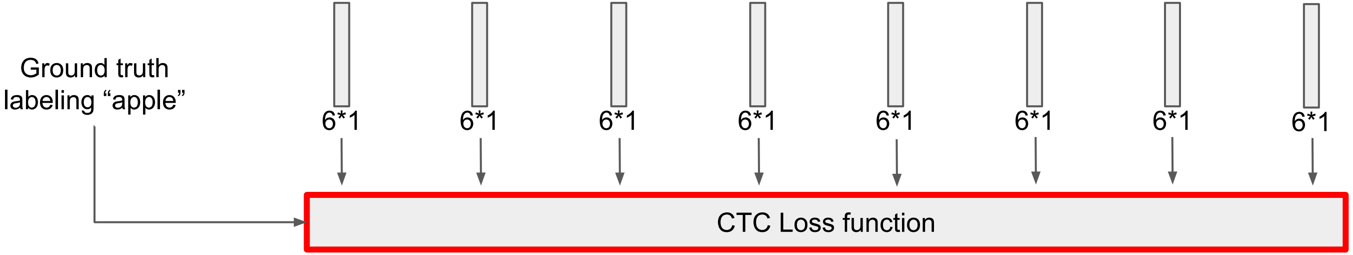

Paths that are correspond to certain label
Set of all possible paths
Set of all possible labelings

path_18
path_19
path_20
path_21
Paths that are correspond to
certain label
labeling_7
Set of all possible paths
Set of all possible labelings

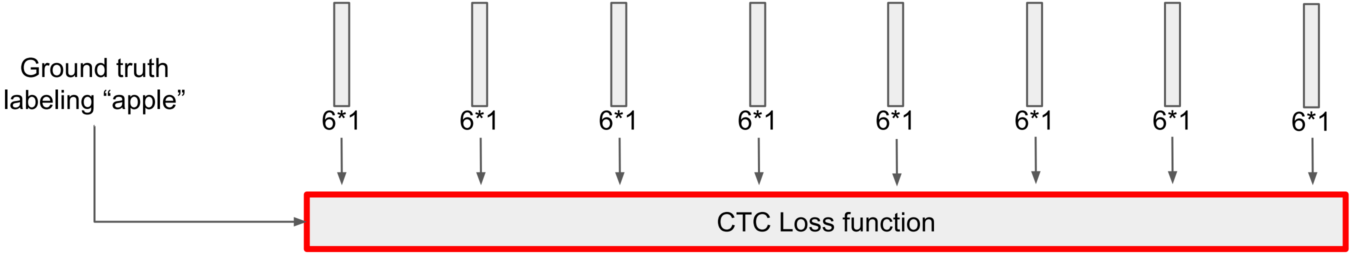

path_18
path_19
path_20
path_21
Paths that are correspond to
certain label
labeling_7
Set of all possible paths
Set of all possible labelings

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


Ground truth
labeling “apple”
6*1
6*1
6*1
6*1
6*1
6*1
6*1
6*1
CTC Loss function

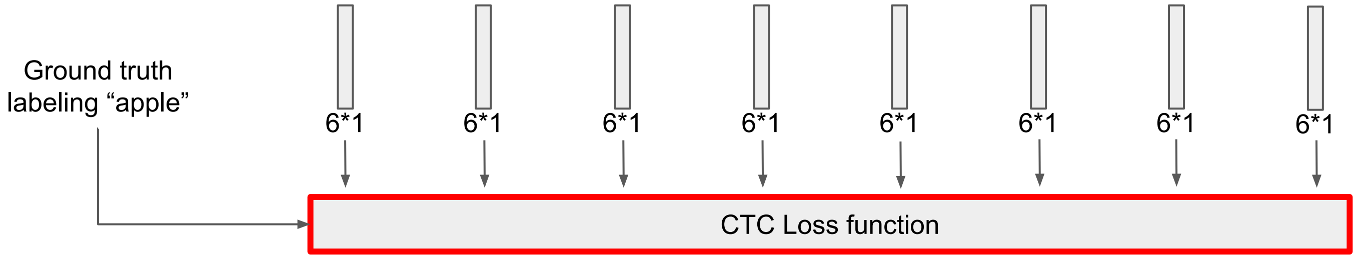


CTC Loss = -ln( p(“apple”) )



CTC Loss = -ln( p(“apple”) )

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



Ground truth labeling “apple”

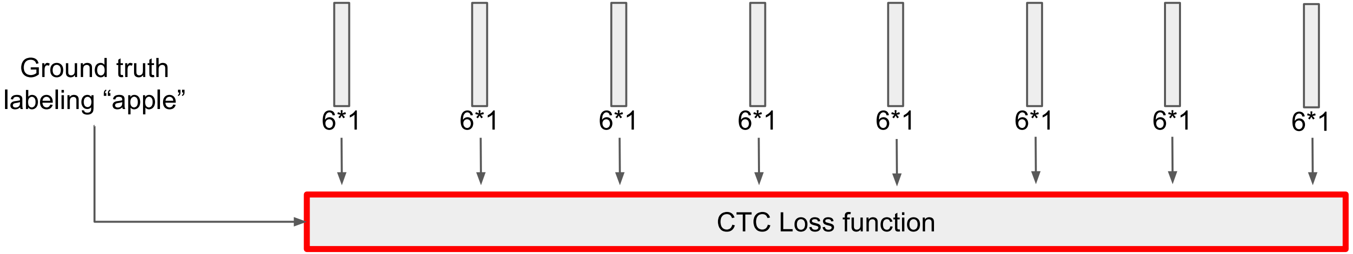
6*1 6*1 6*1 6*1 6*1 6*1 6*1 6*1

CTC Loss function

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.



Ground truth labeling “apple”

6*1 6*1 6*1 6*1 6*1 6*1 6*1 6*1

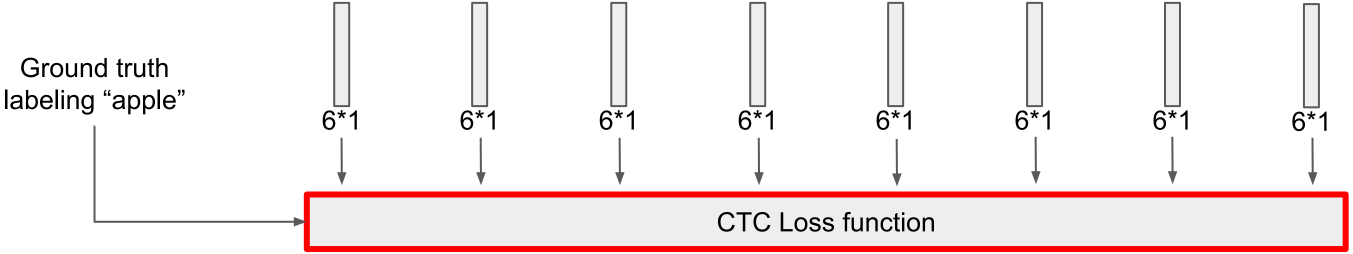
CTC Loss function

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.



Ground truth labeling “apple”

6*1 6*1 6*1 6*1 6*1 6*1 6*1 6*1

CTC Loss function

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

Let’s try to find all paths, that are correspond to certain labeling “apple” using dynamic programming.

Construct such table:

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

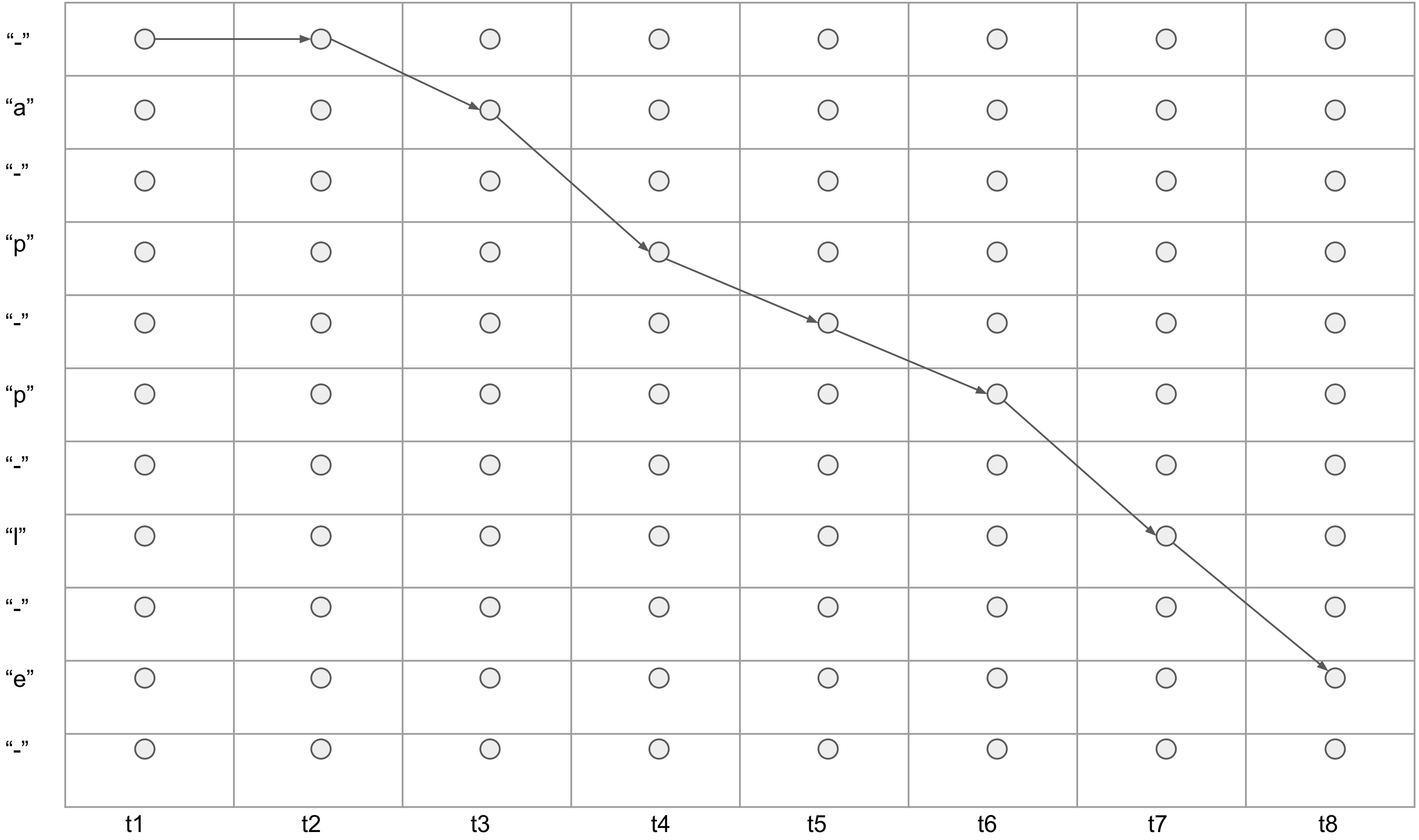
Few words about what this table means.

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

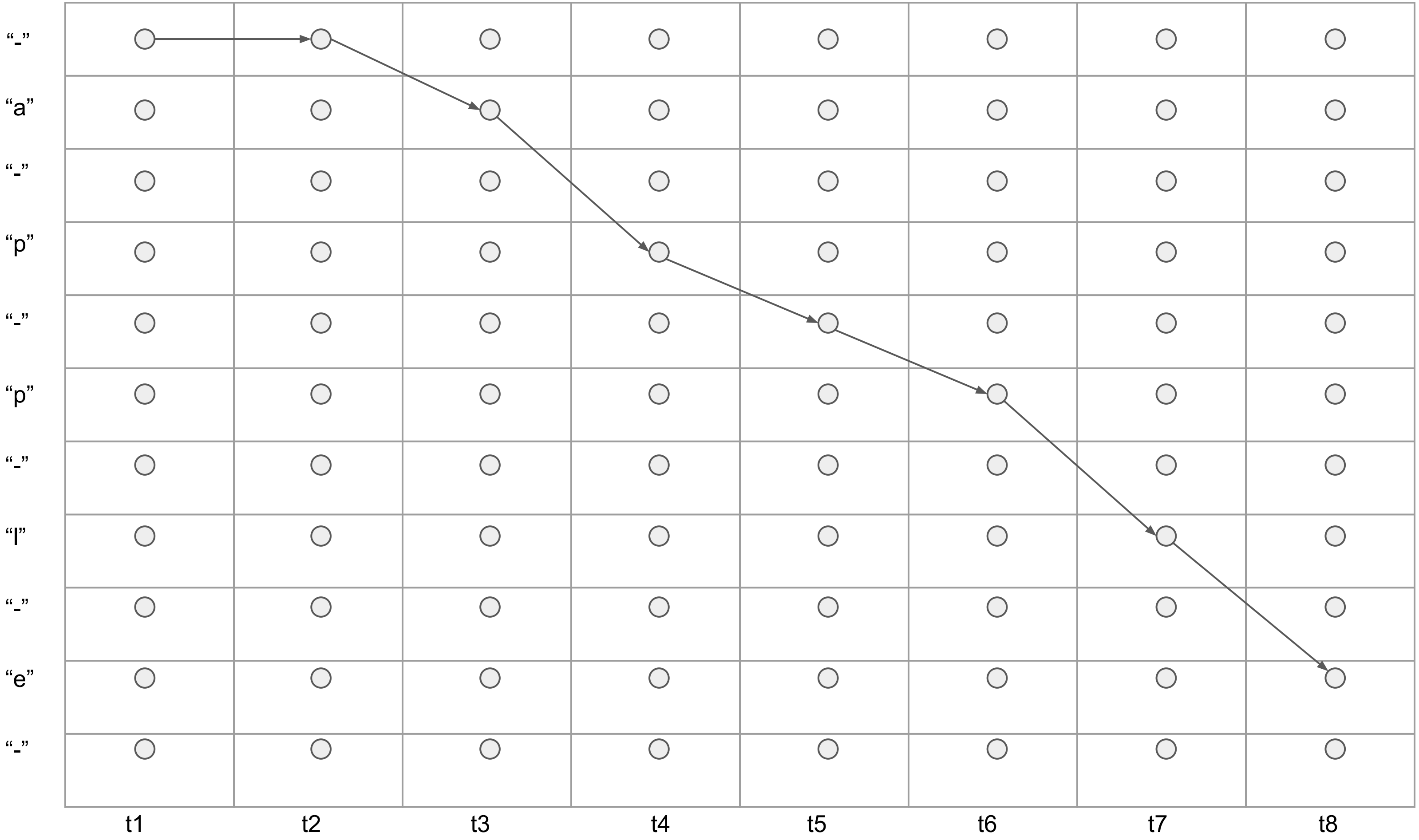
With this table we can model all paths that are correspond to ground truth labeling “apple”

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

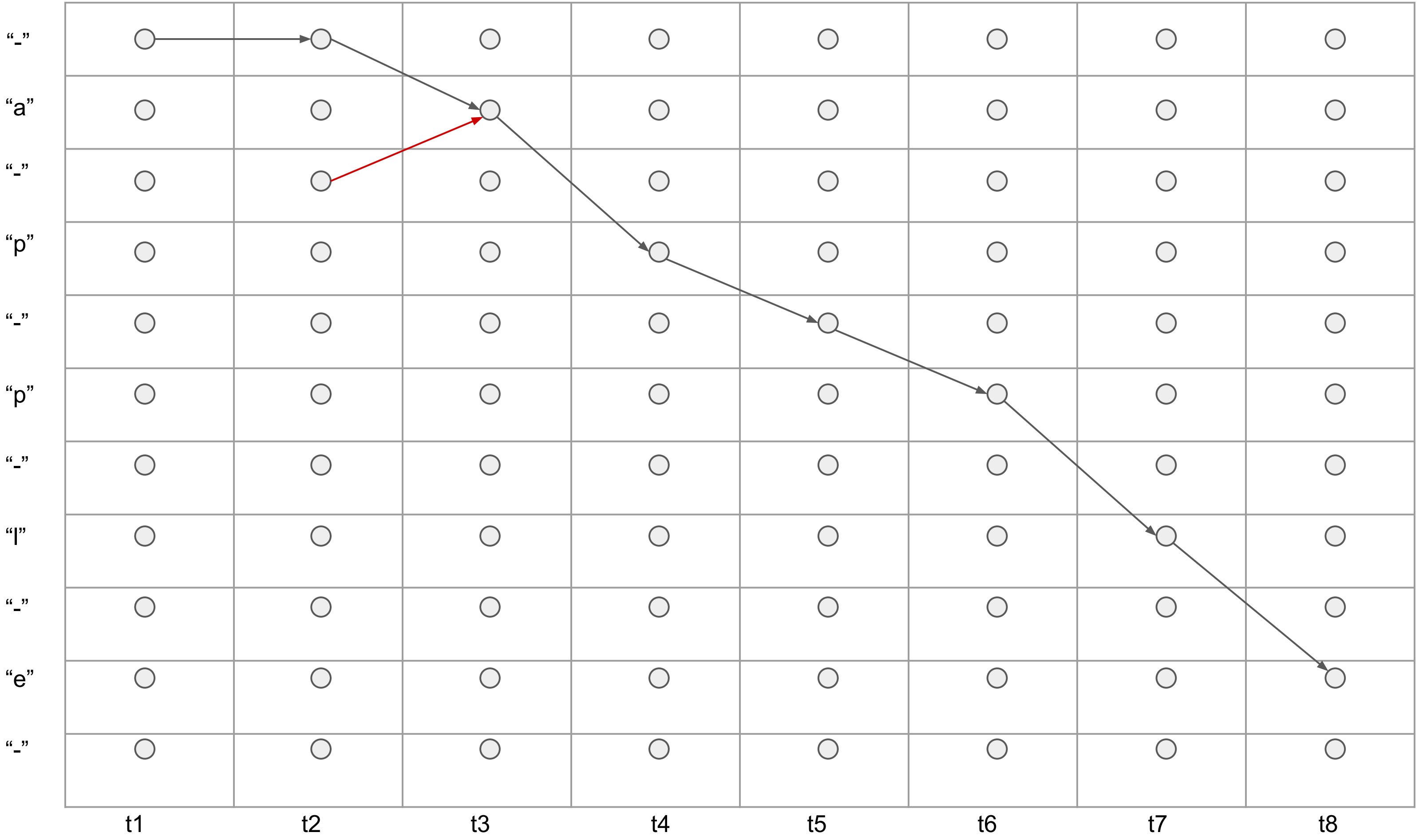
For example: path “--ap-ple” can be mapped to labeling “apple”. i.e: B(“--ap-ple”) = “apple”



Arrows can not end in upper node.

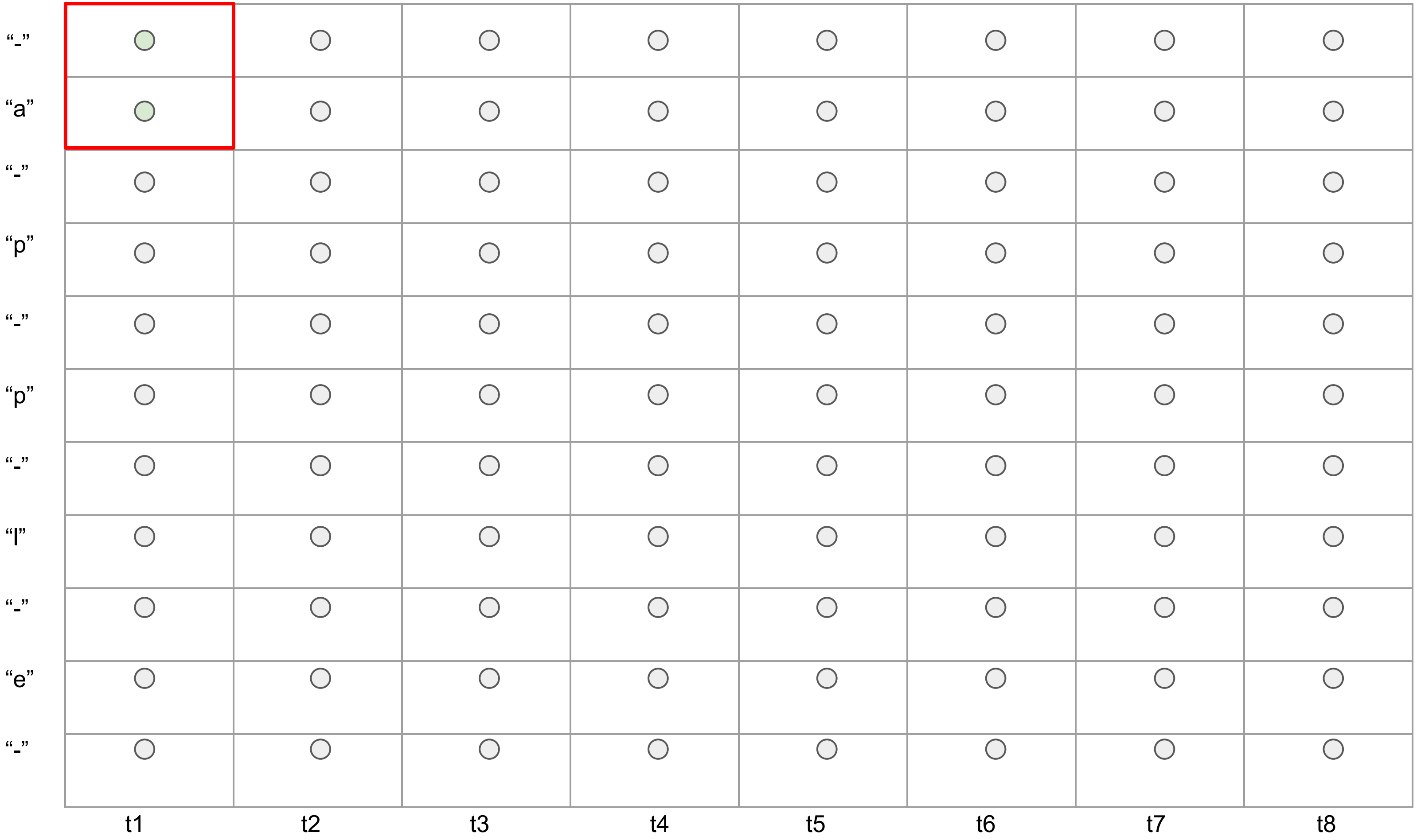


Example: the red transition is impossible, because we can not predict “-” (third symbol in sequence) at time2 and then predict “a” (second symbol in sequence) at time 3.

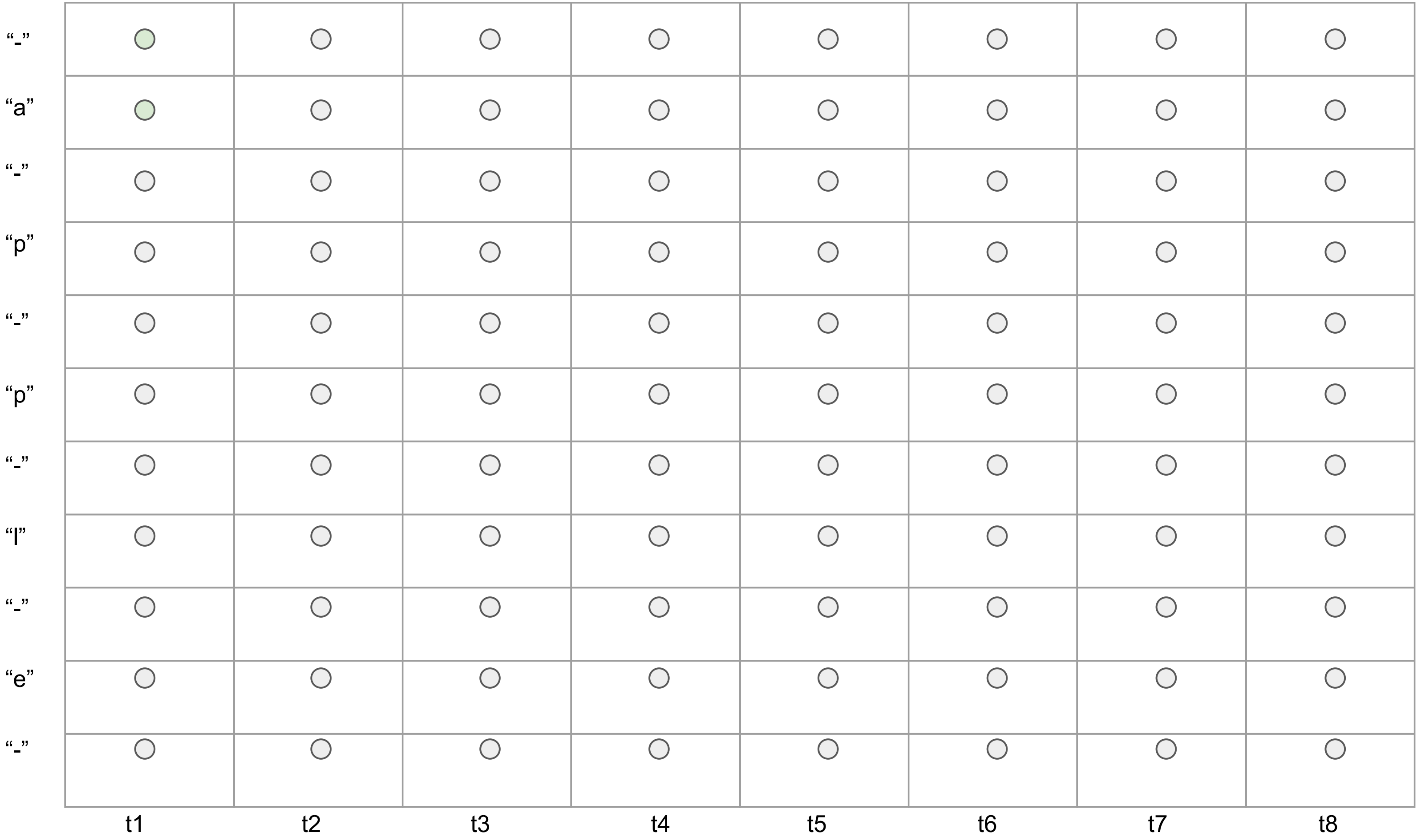


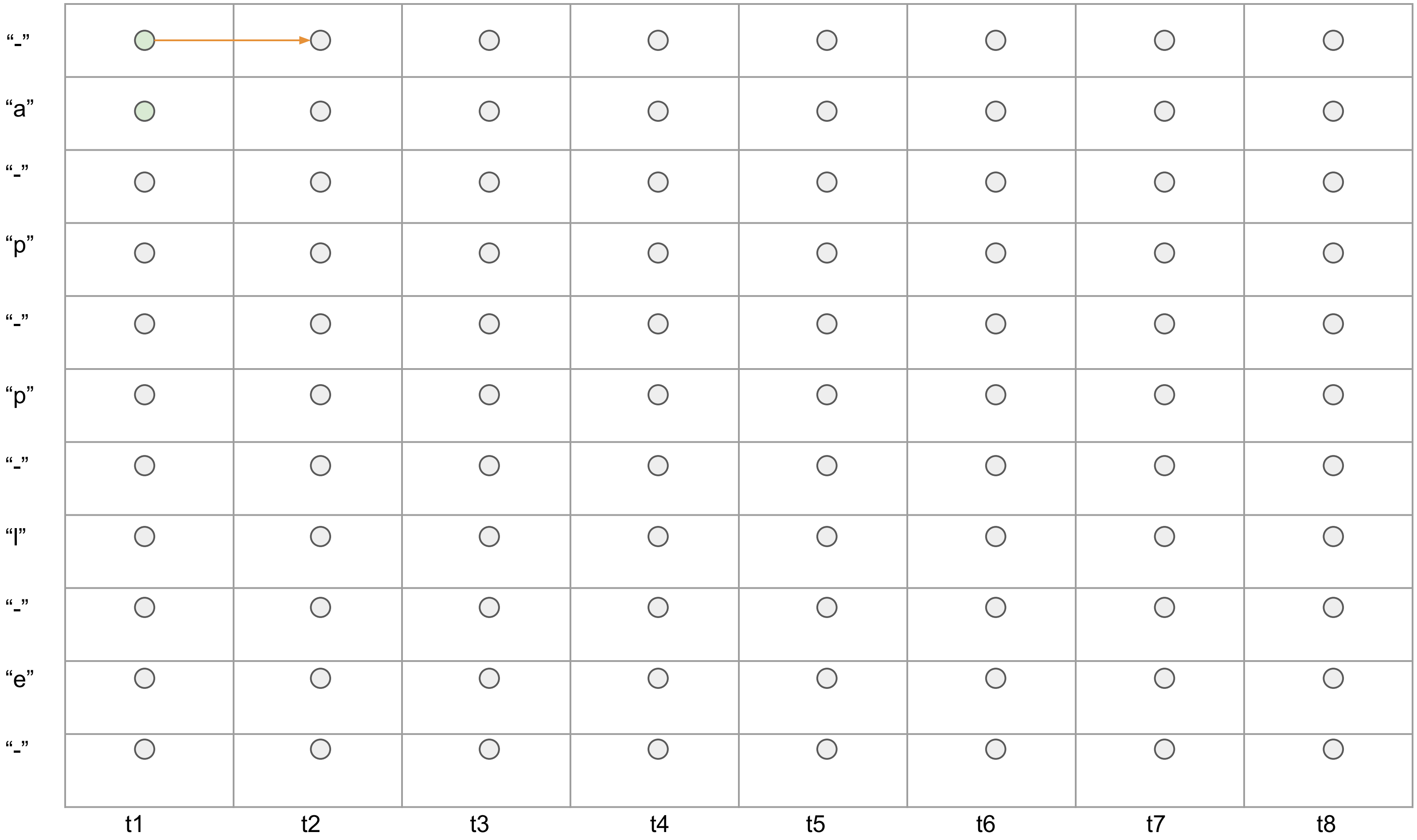
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| “-” | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| “a” | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| “-” | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| “p” | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| “-” | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| “p” | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| “-”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

Initialization: paths can start only with this symbols.

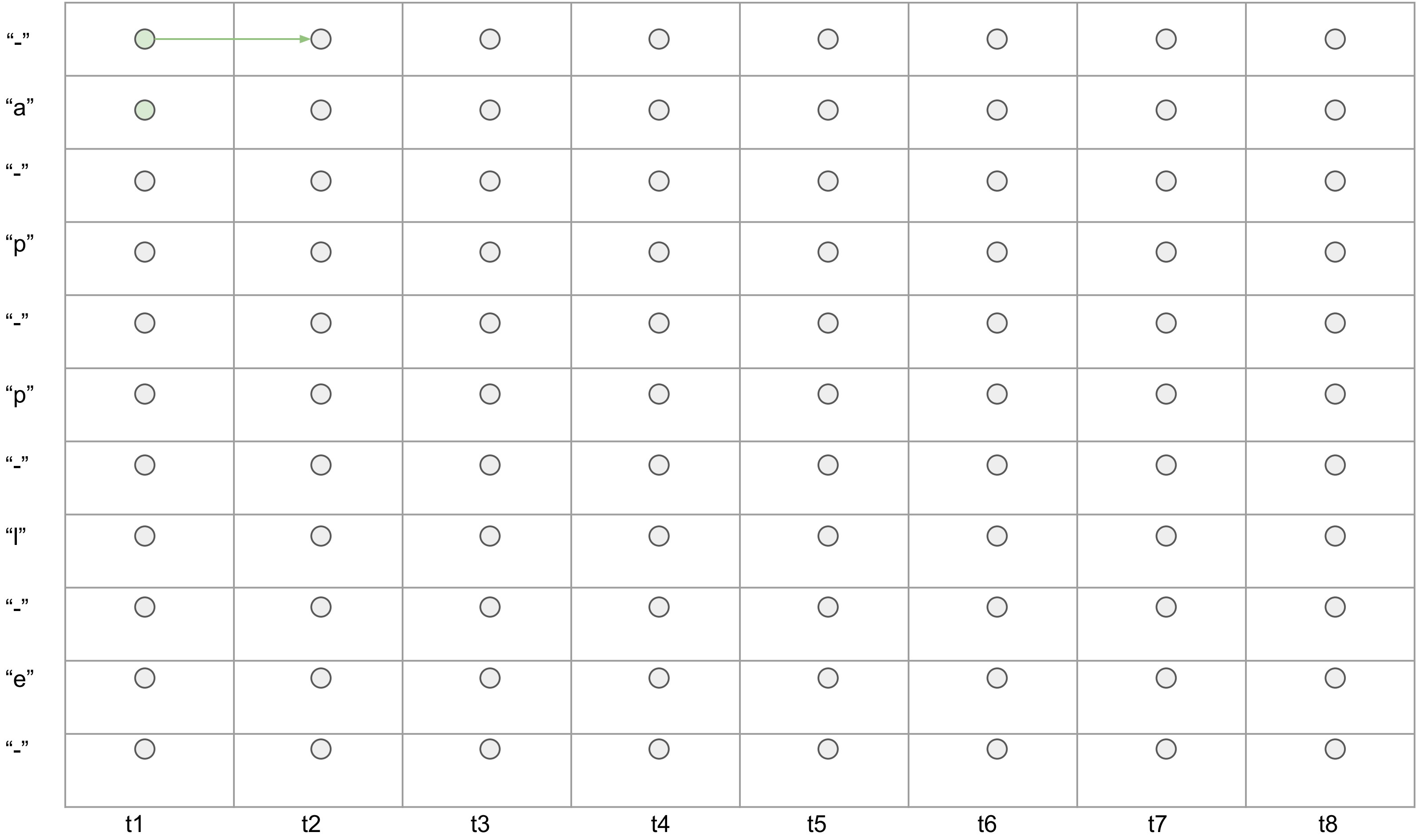


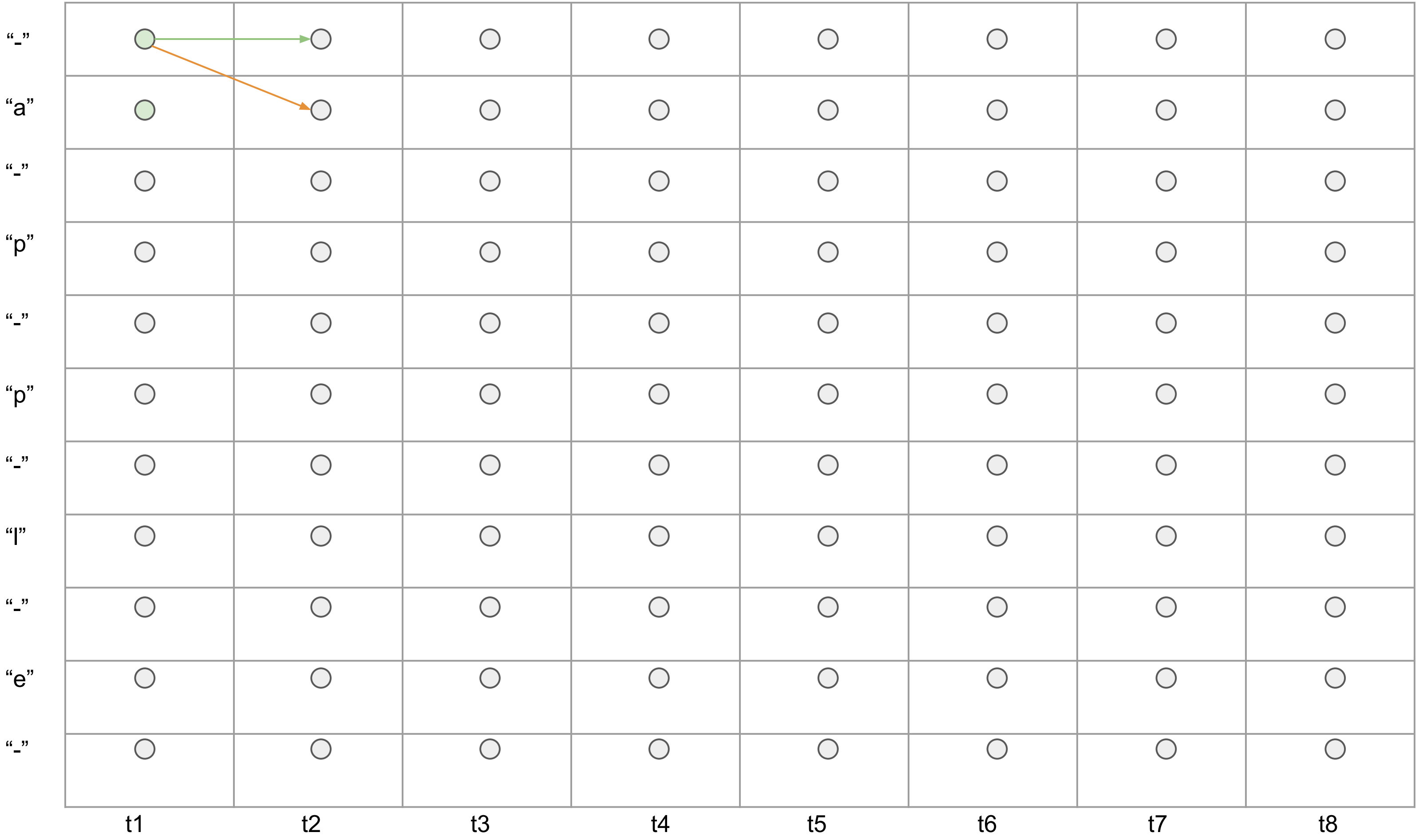
Let’s consider possible transitions from these two start points.



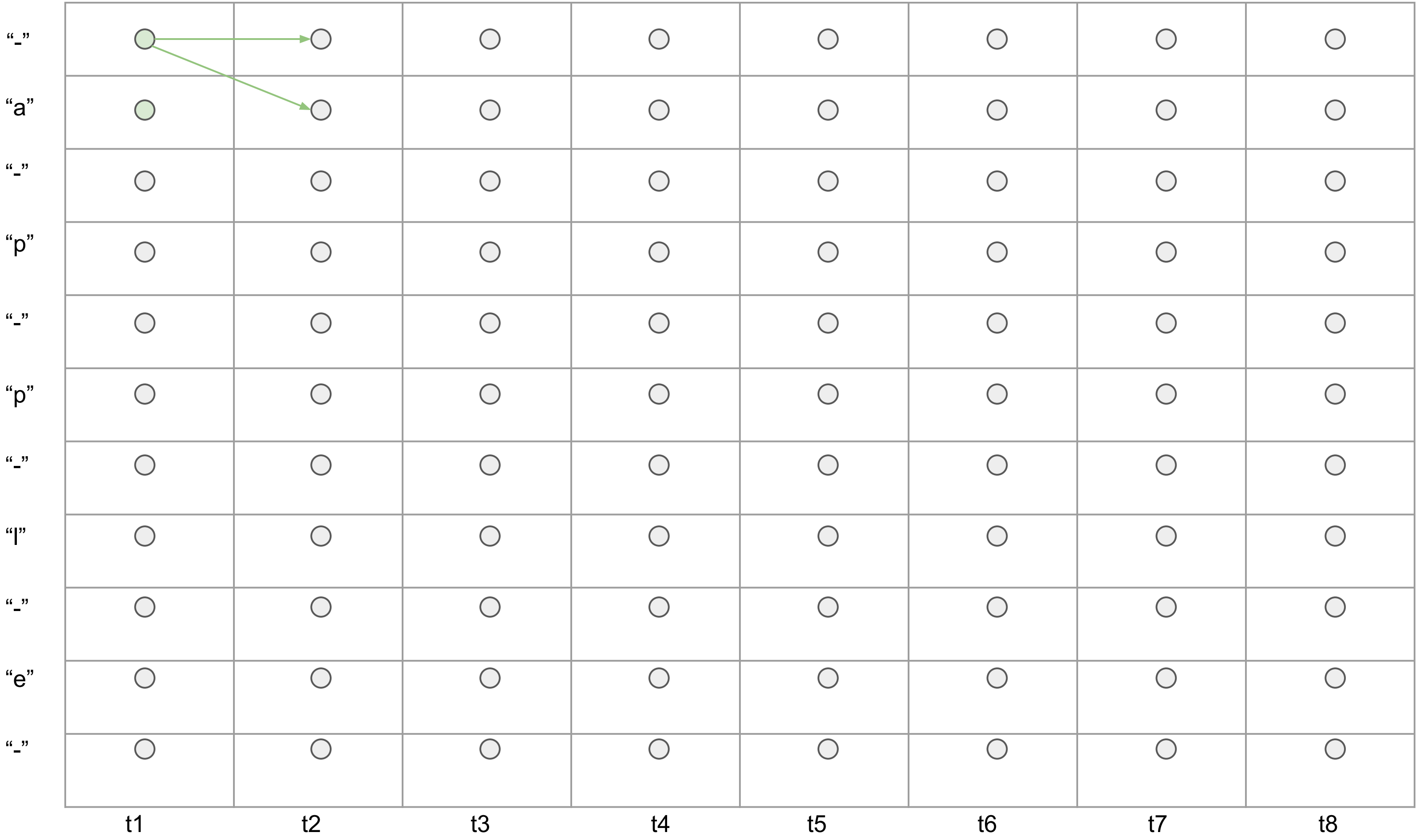

“-”
“a”
“-”
“p”
“-”
“p”
“-”
“l”
“-”
“e”
“-”
t1
t2
t3
t4
t5
t6
t7
t8

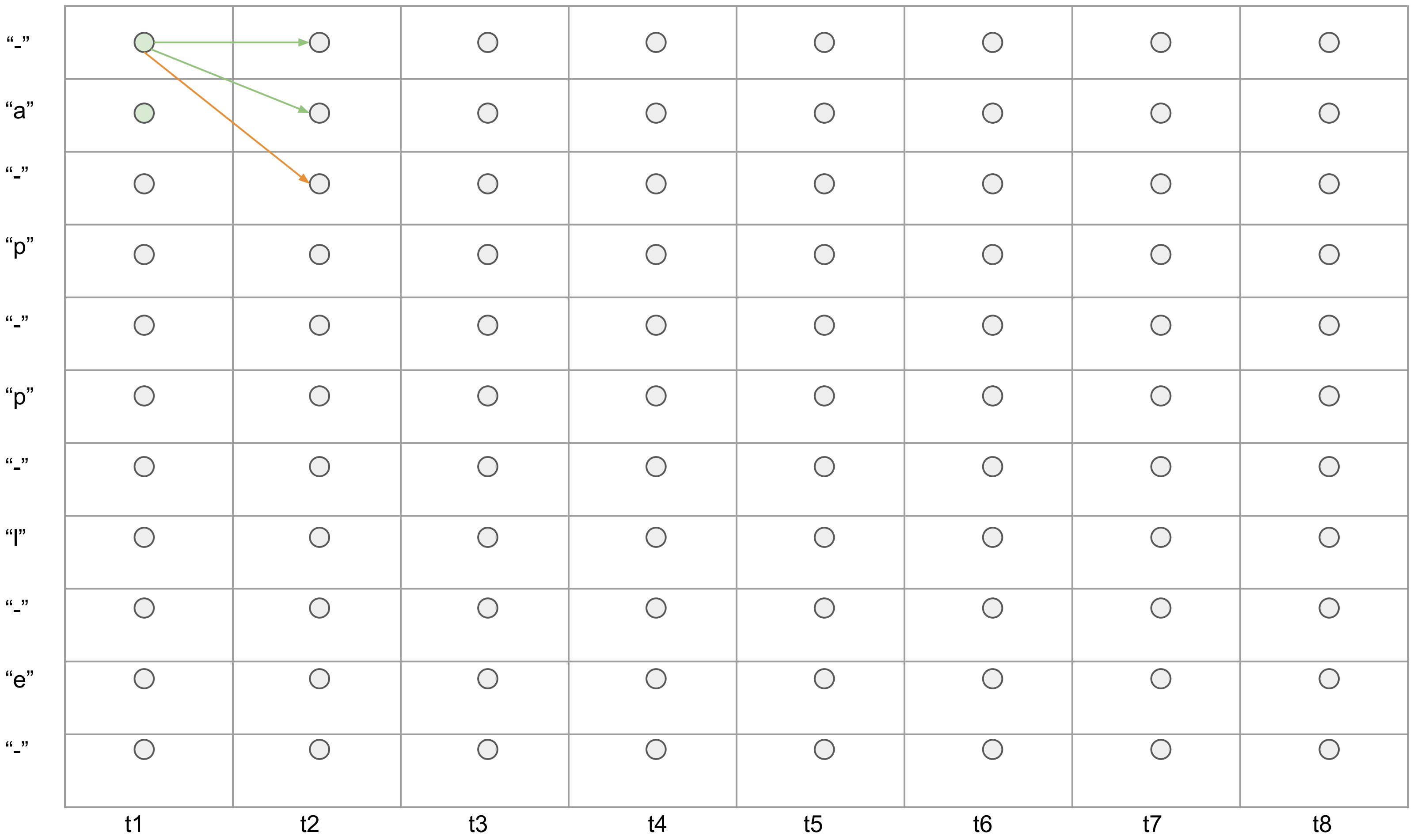
Possible transition, because paths can start with two blanks. Example of valid path: “--ap-ple”. B(“--ap-ple”)=“apple”



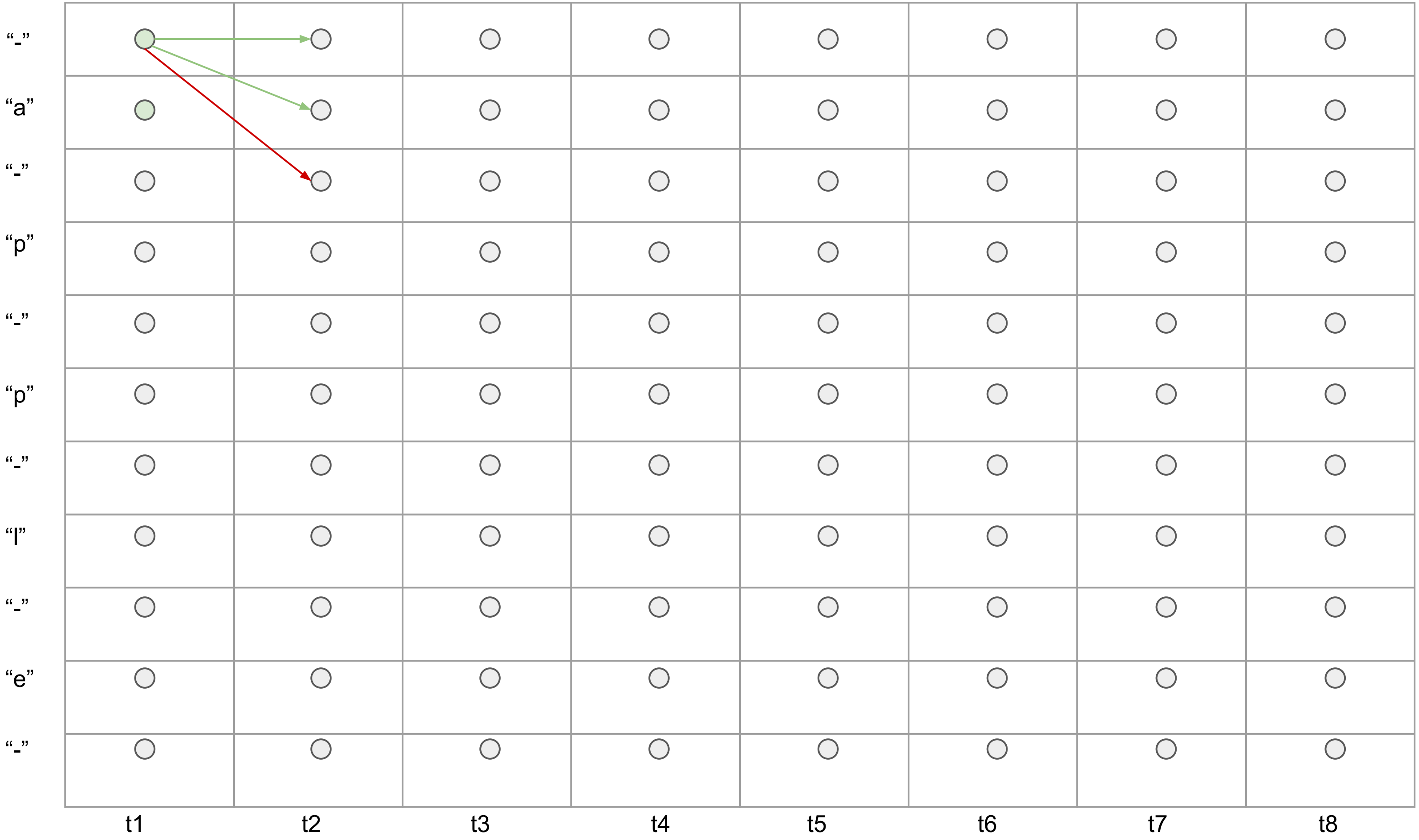

"-"
"a"
"-"
"p"
"-"
"p"
"-"
"l"
"-"
"e"
"-"
t1
t2
t3
t4
t5
t6
t7
t8

Possible transition, because paths can start with “-a”. Example of valid path: “-aap-ple”. B(“-aap-ple”)=“apple”

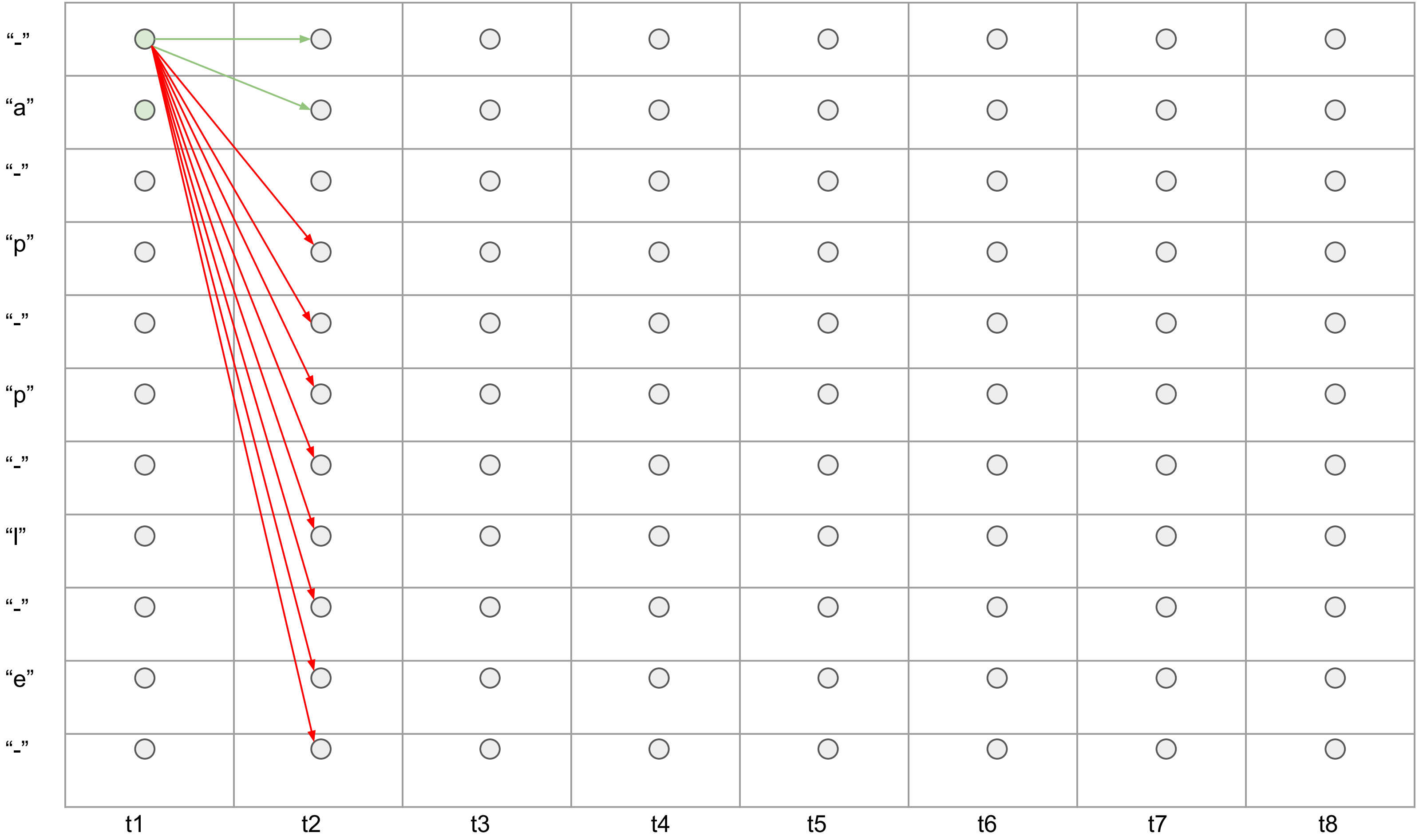


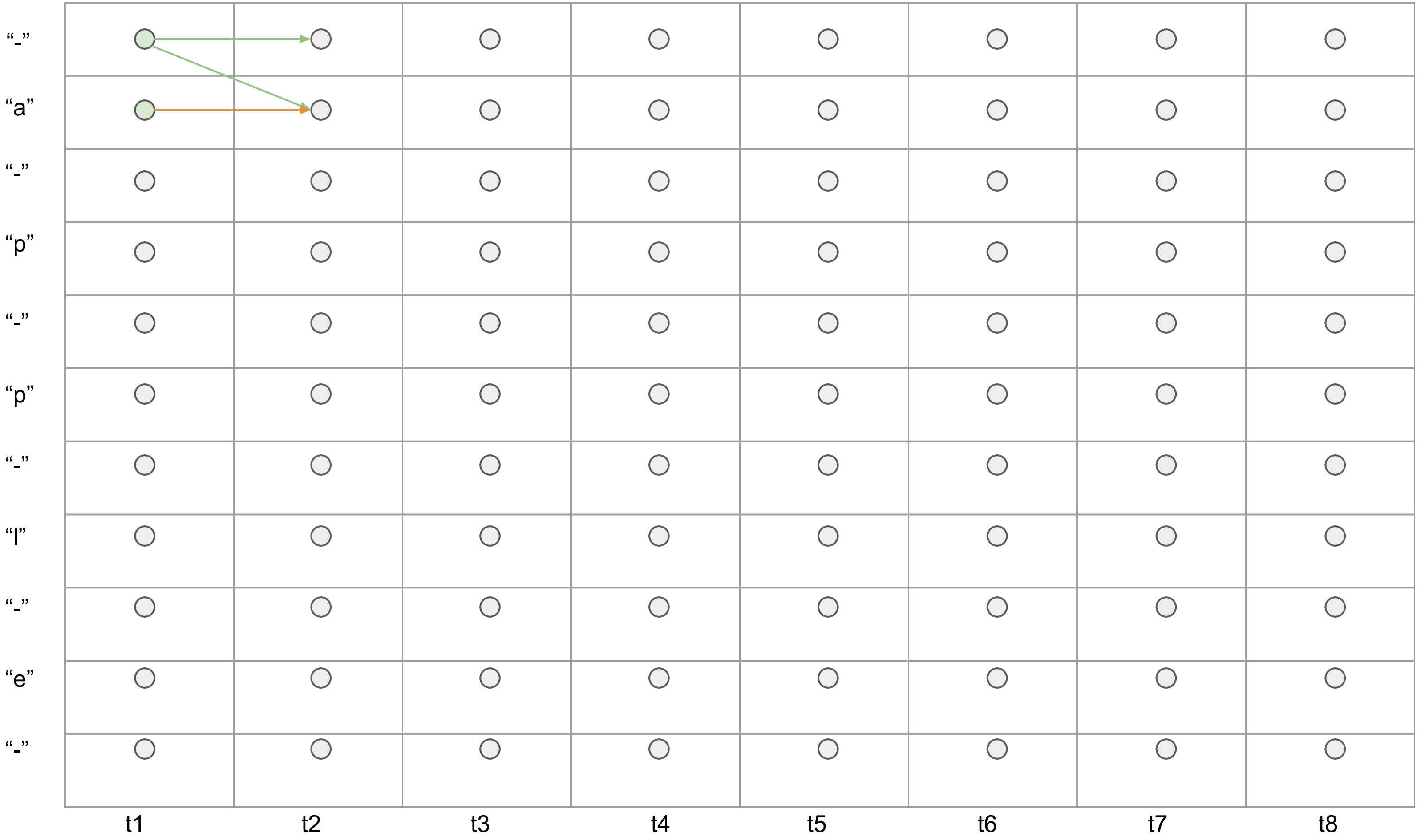

“-”
“a”
“-”
“p”
“-”
“p”
“-”
“l”
“-”
“e”
“-”
t1
t2
t3
t4
t5
t6
t7
t8

Impossible transition, because we can not skip symbol “a”. Then we will see, that we can skip only blanks.

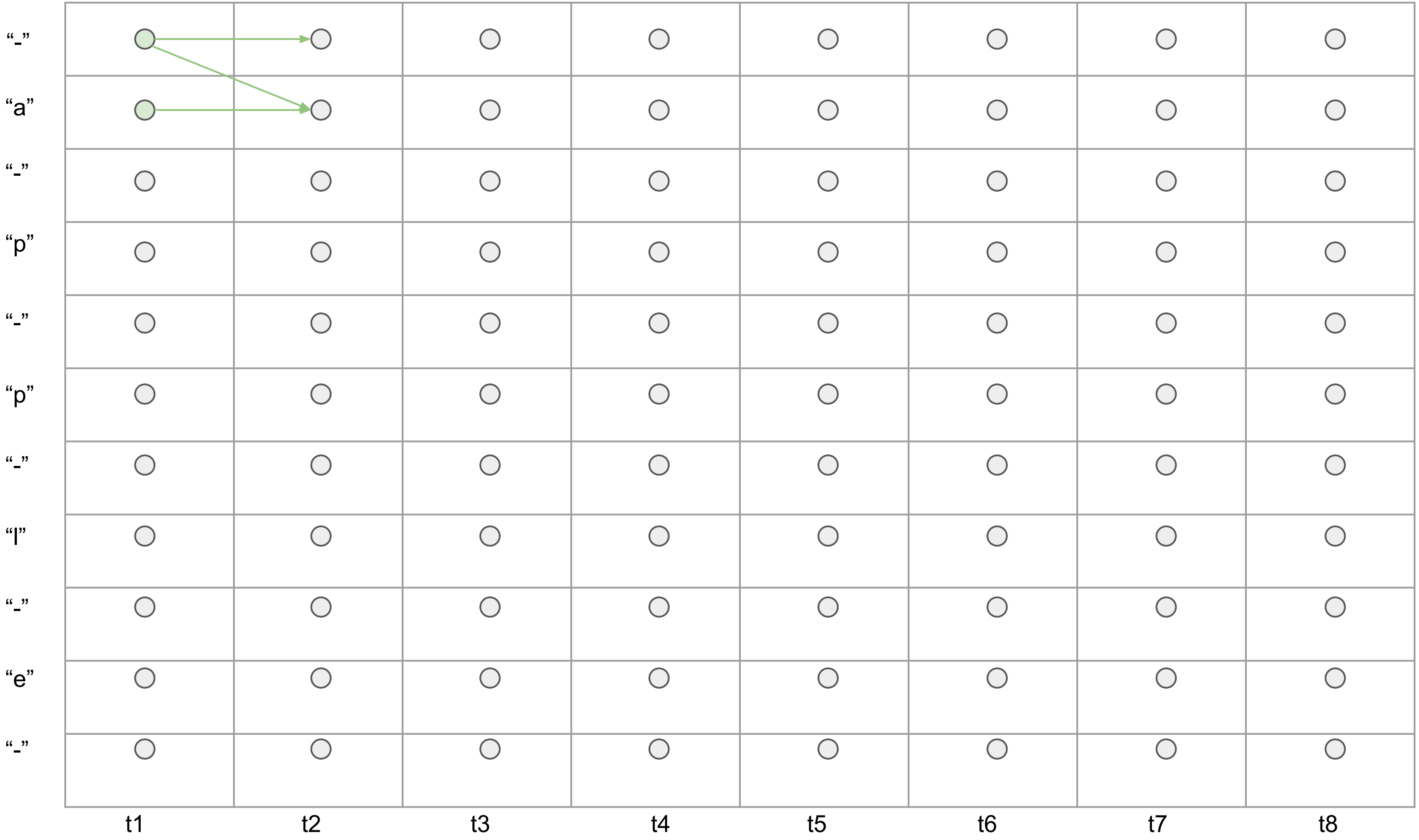


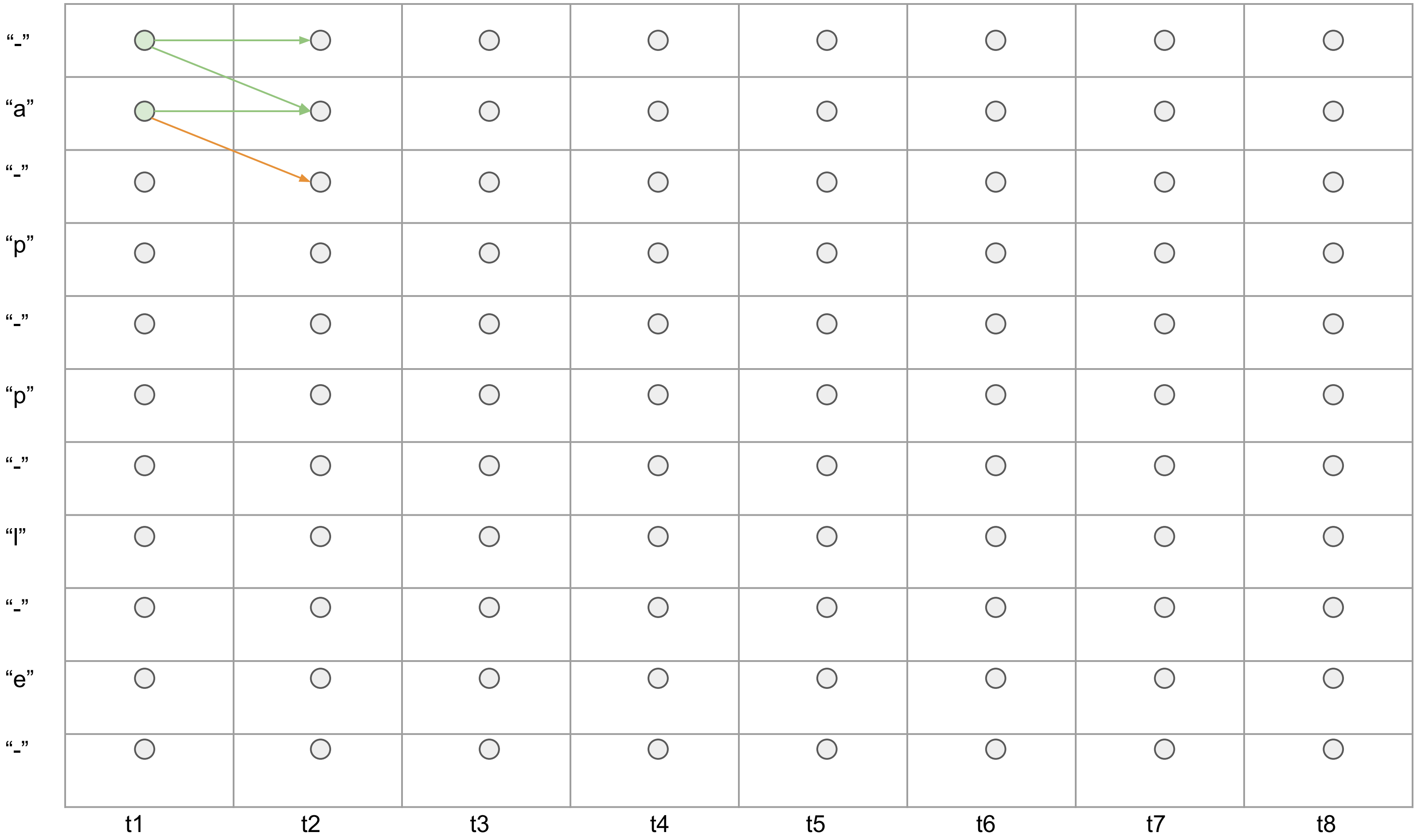
The red transitions are also impossible by the same logic.



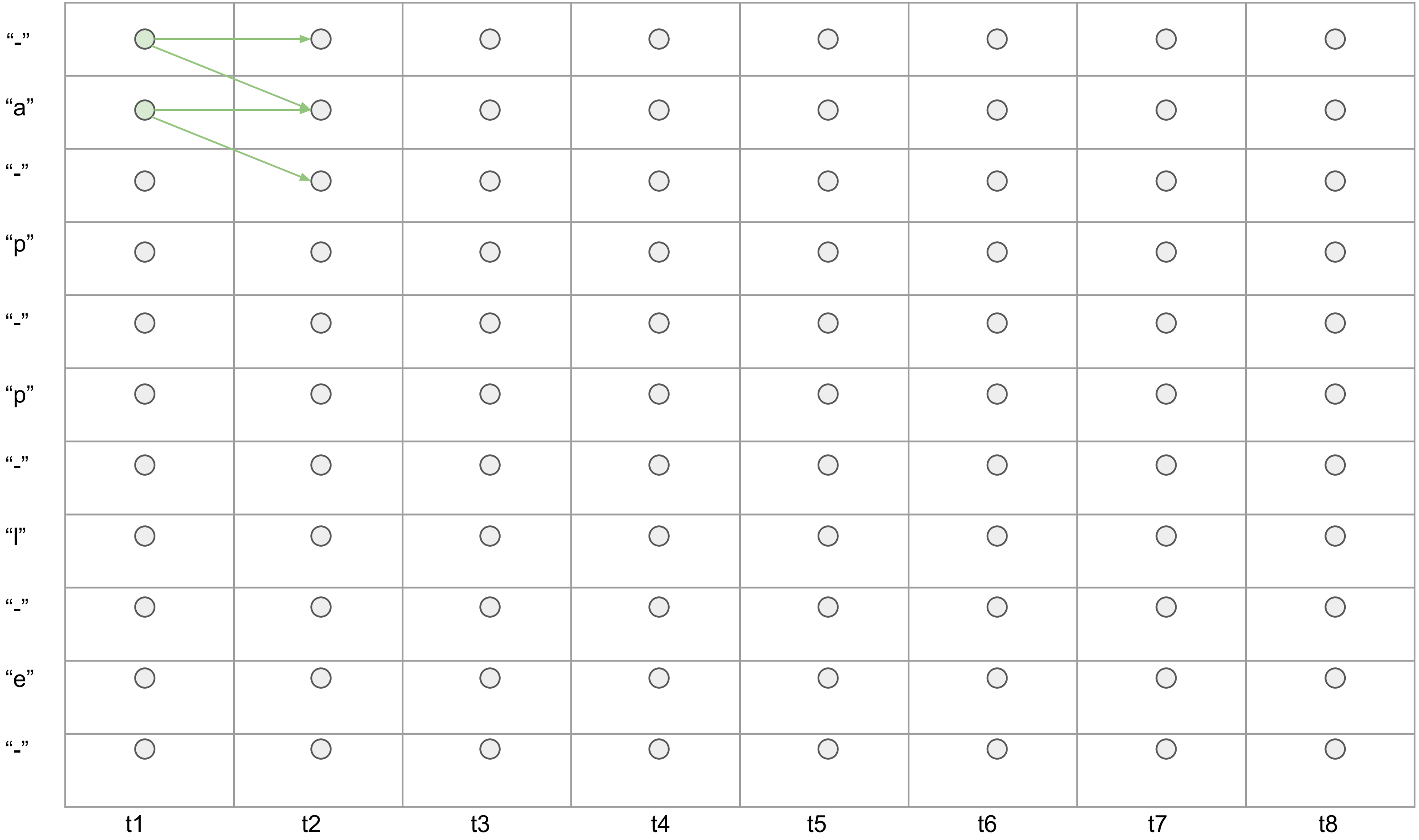

“-”
“a”
“-”
“p”
“-”
“p”
“-”
“l”
“-”
“e”
“-”
t1
t2
t3
t4
t5
t6
t7
t8

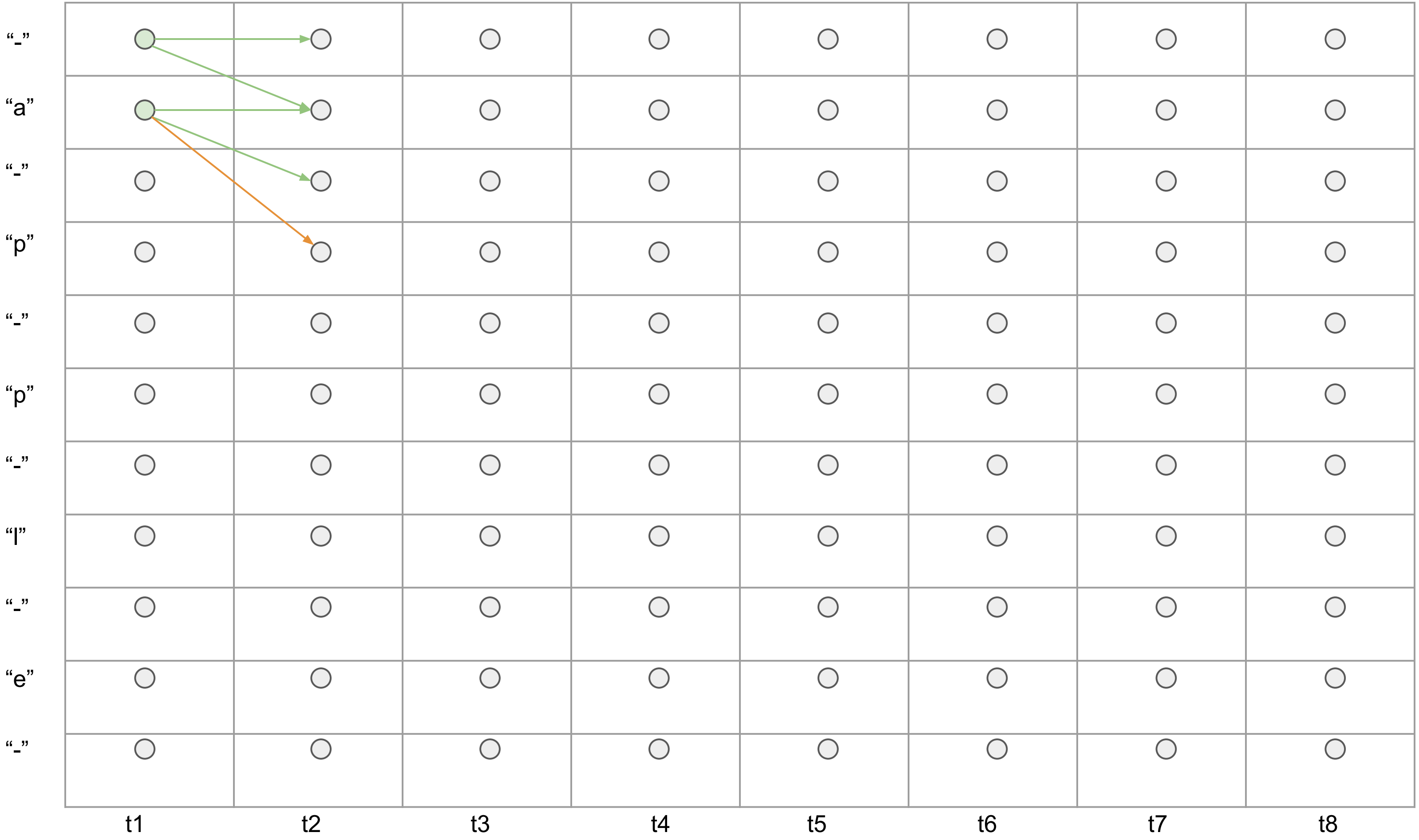
Possible transition, because paths can start with “aa”. Example of valid path: “aa-p-ple”. B(“aa-p-ple”)=“apple”



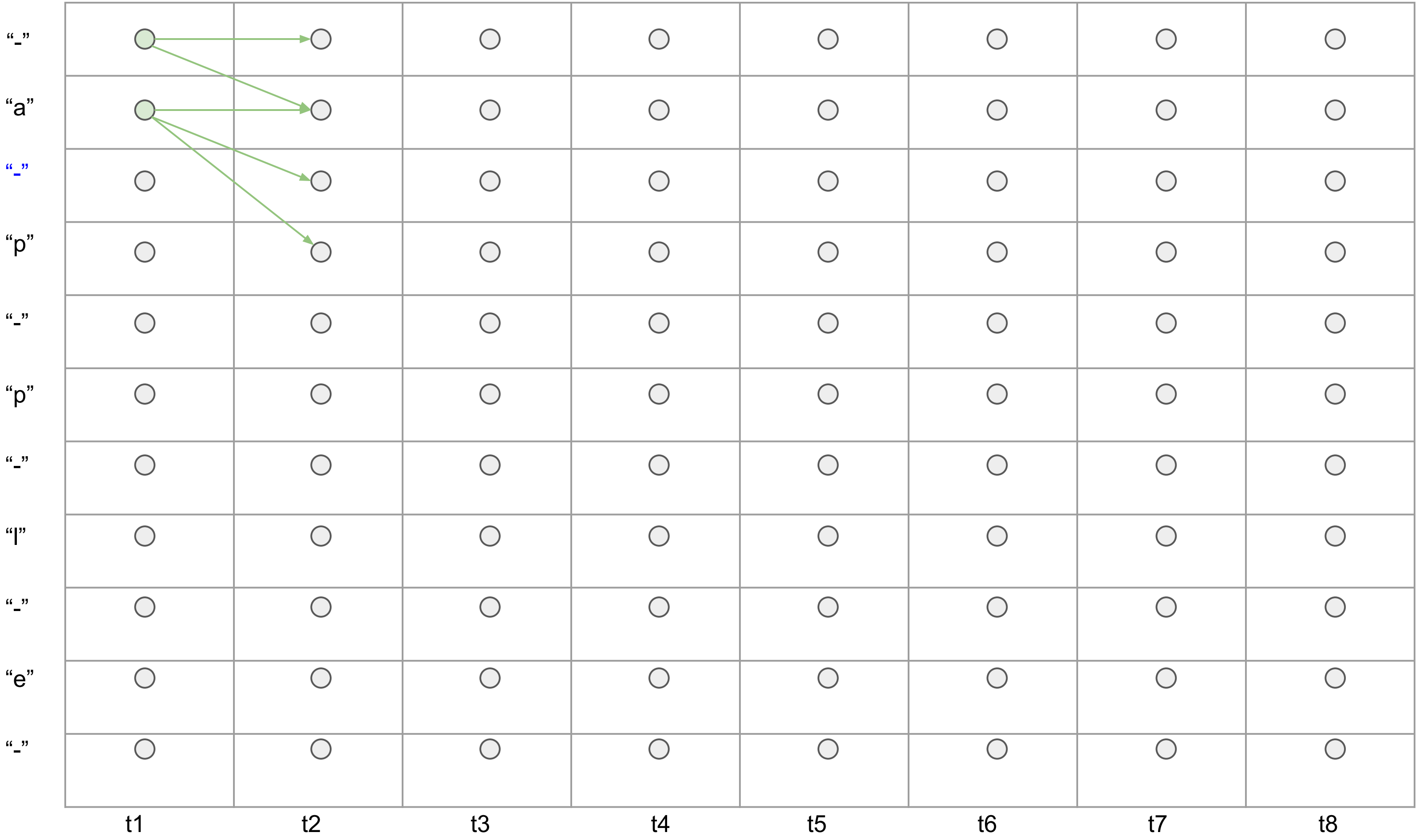

“-”
“a”
“-”
“p”
“-”
“p”
“-”
“l”
“-”
“e”
“-”
t1
t2
t3
t4
t5
t6
t7
t8

Possible transition, because paths can start with “a-”. Example of valid path: “a--p-ple”. B(“a--p-ple”)=“apple”

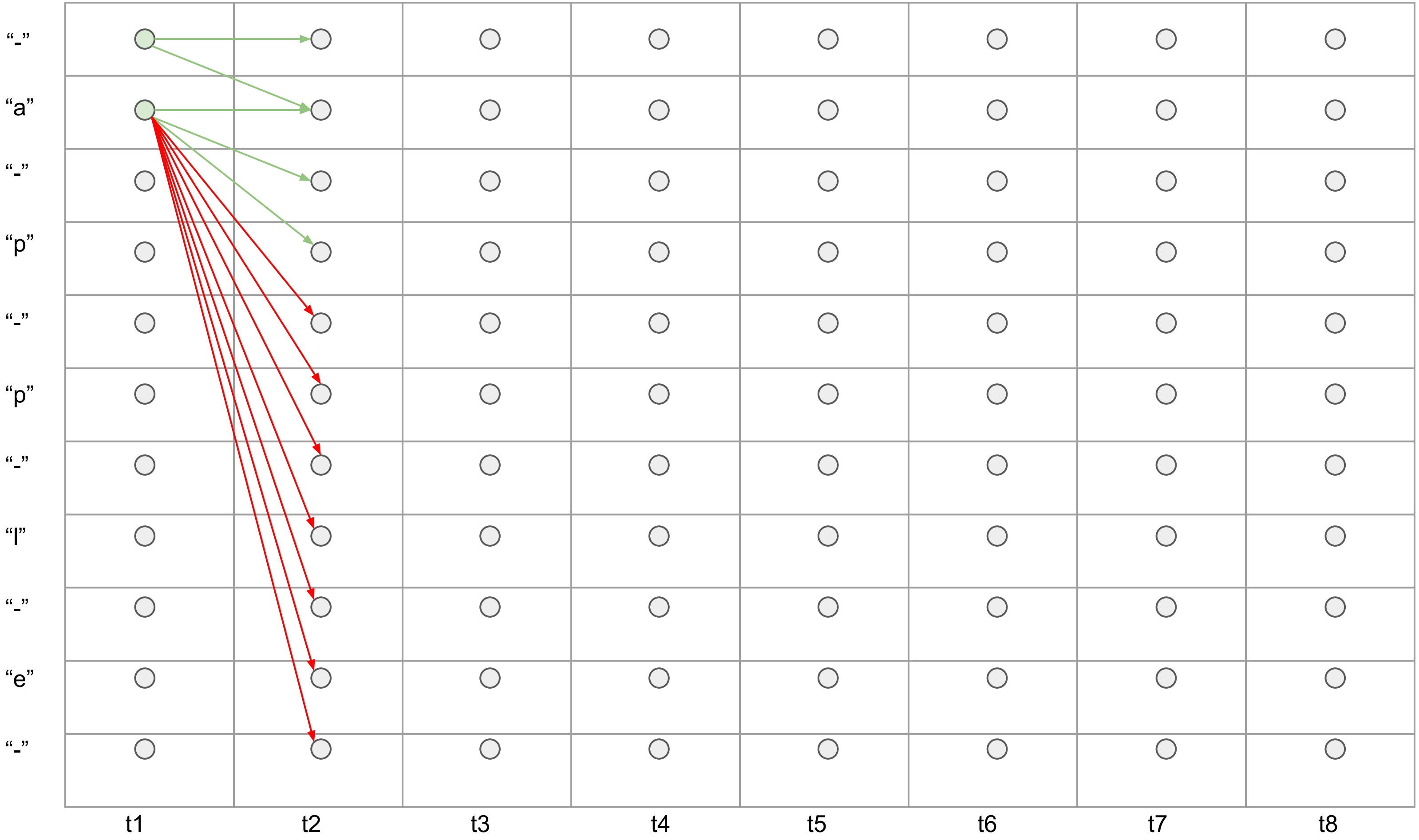


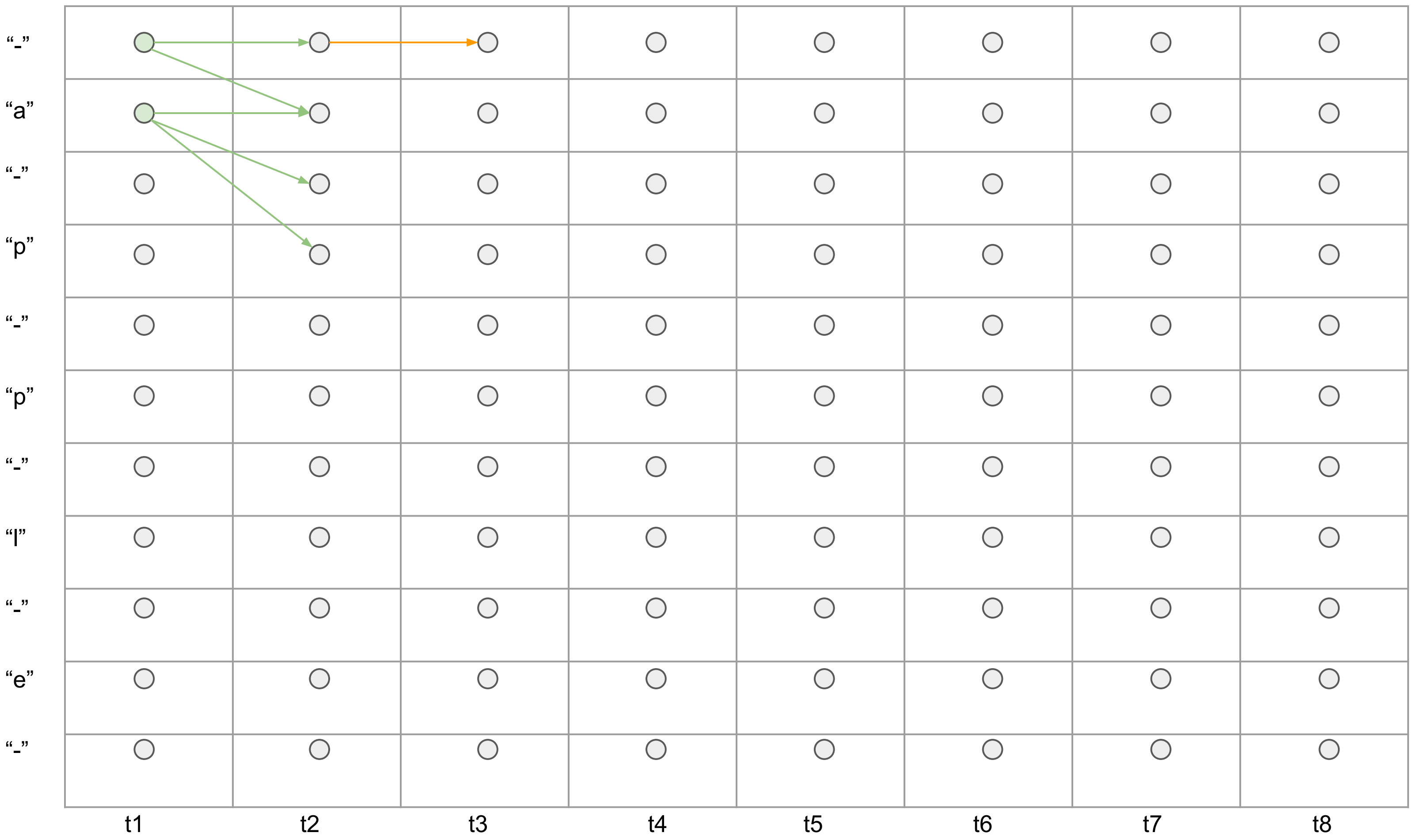

"-"
"a"
"-"
"p"
"-"
"p"
"-"
"l"
"-"
"e"
"-"
t1
t2
t3
t4
t5
t6
t7
t8

Possible transition, because paths can start with “ap”. Example of valid path: “app-pl-e”. B(“app-pl-e”)=“apple”. This is the example, where we skip blank (third symbol in sequence)

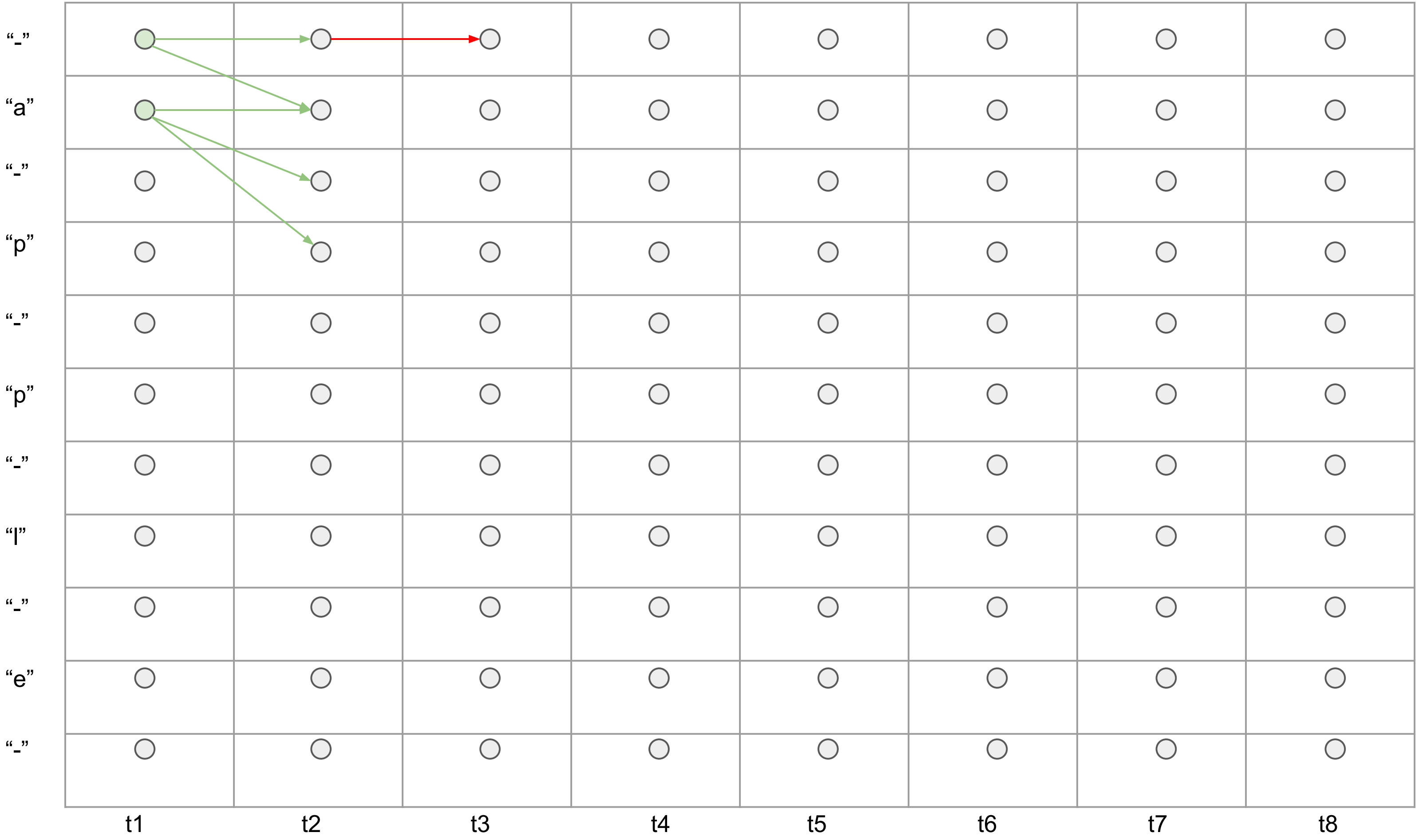


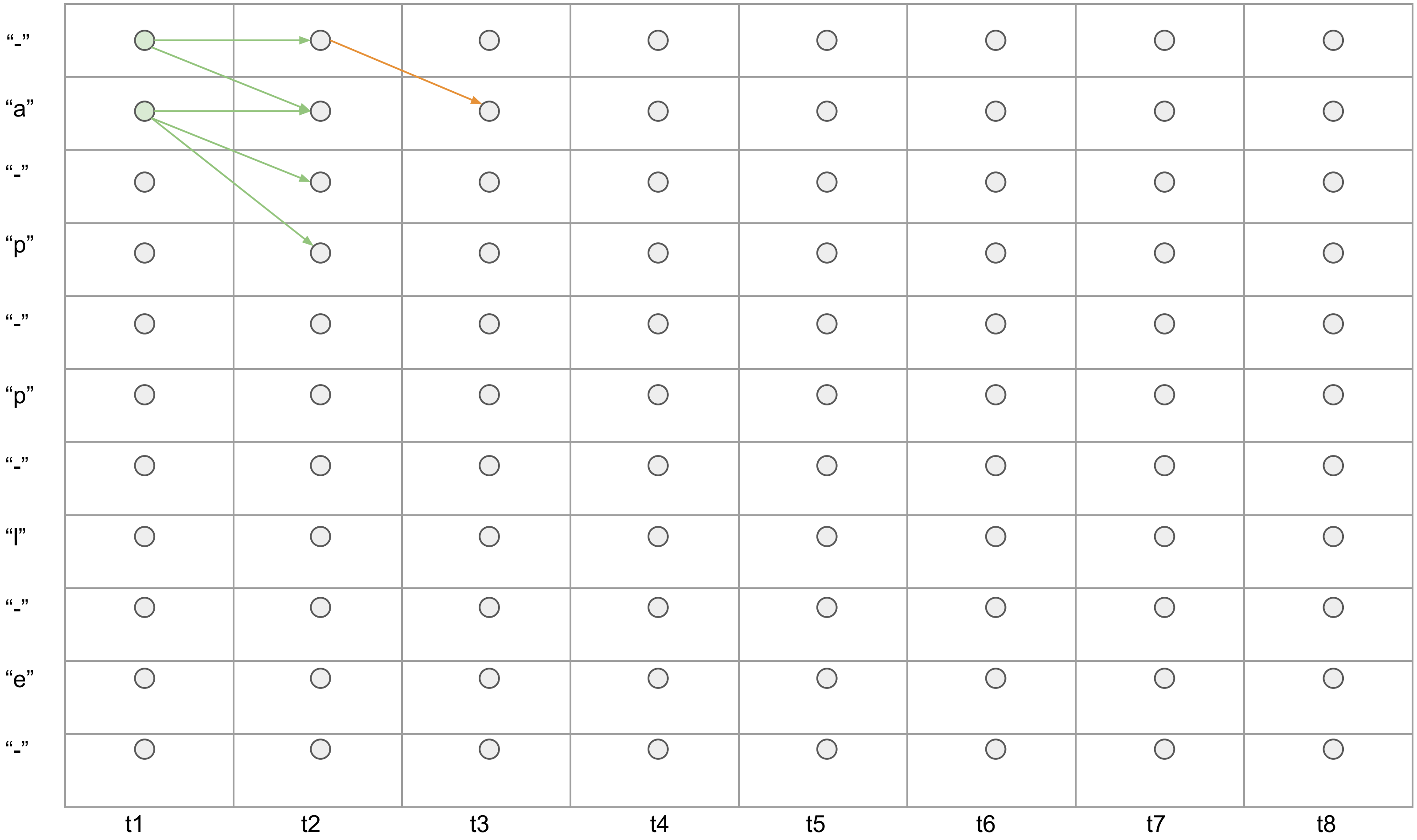
Impossible transitions.



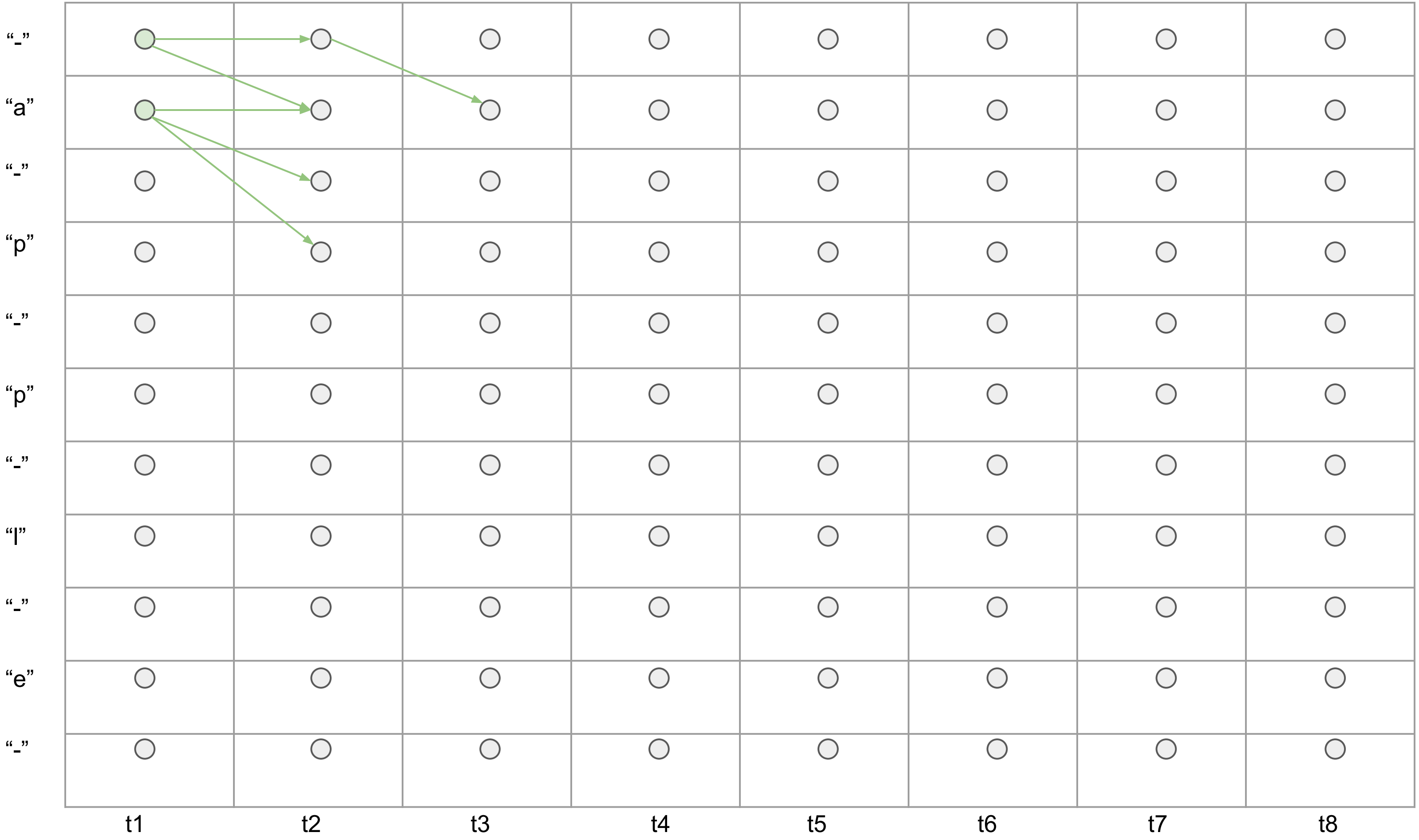

“-”
“a”
“-”
“p”
“-”
“p”
“-”
“l”
“-”
“e”
“-”
t1
t2
t3
t4
t5
t6
t7
t8

Impossible transition, because there are no valid paths that are start with “---”.

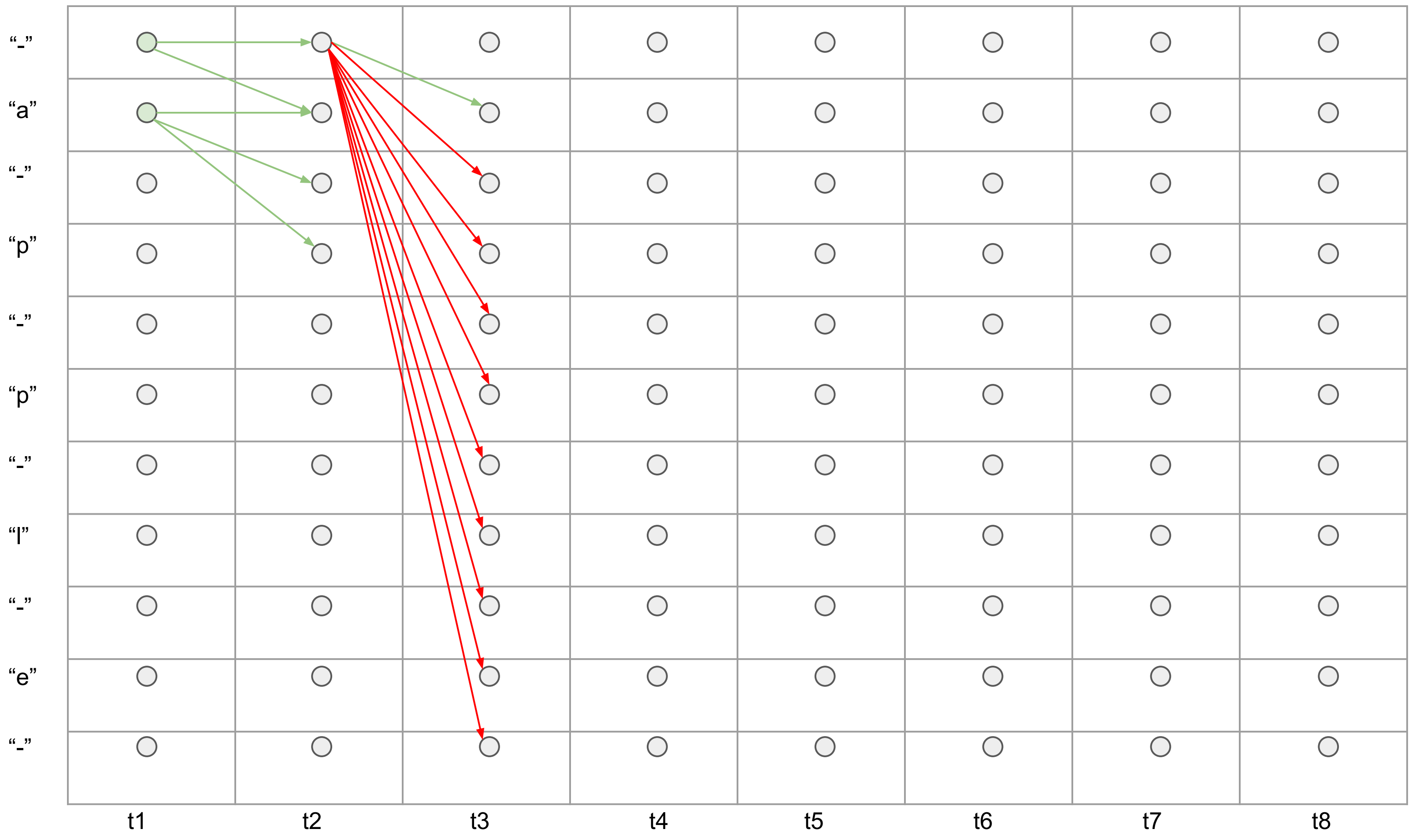


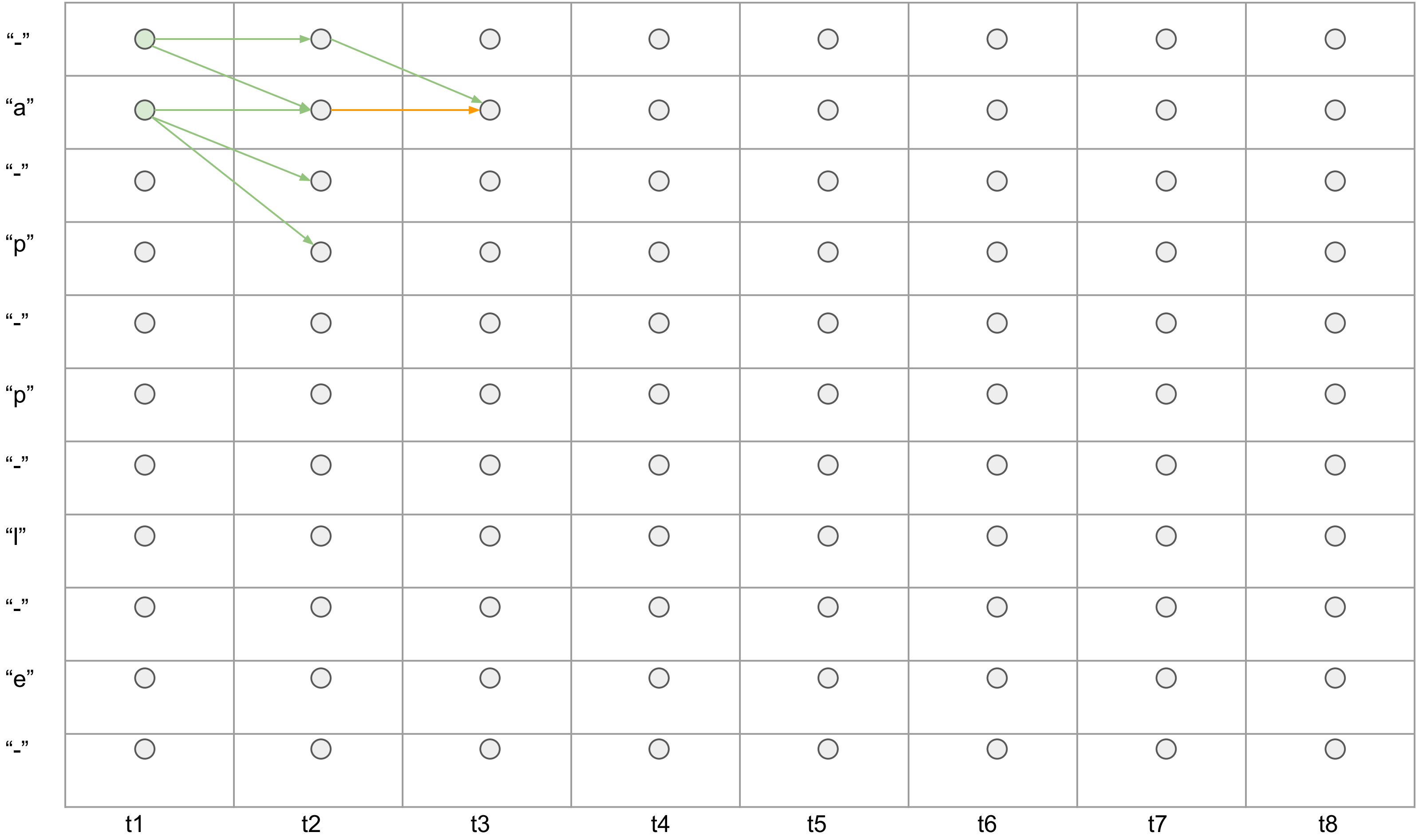

“-”
“a”
“-”
“p”
“-”
“p”
“-”
“l”
“-”
“e”
“-”
t1
t2
t3
t4
t5
t6
t7
t8

Possible transition, because paths can start with “-a”. Example of valid path: “--ap-ple”. B(“--ap-ple”)=“apple”

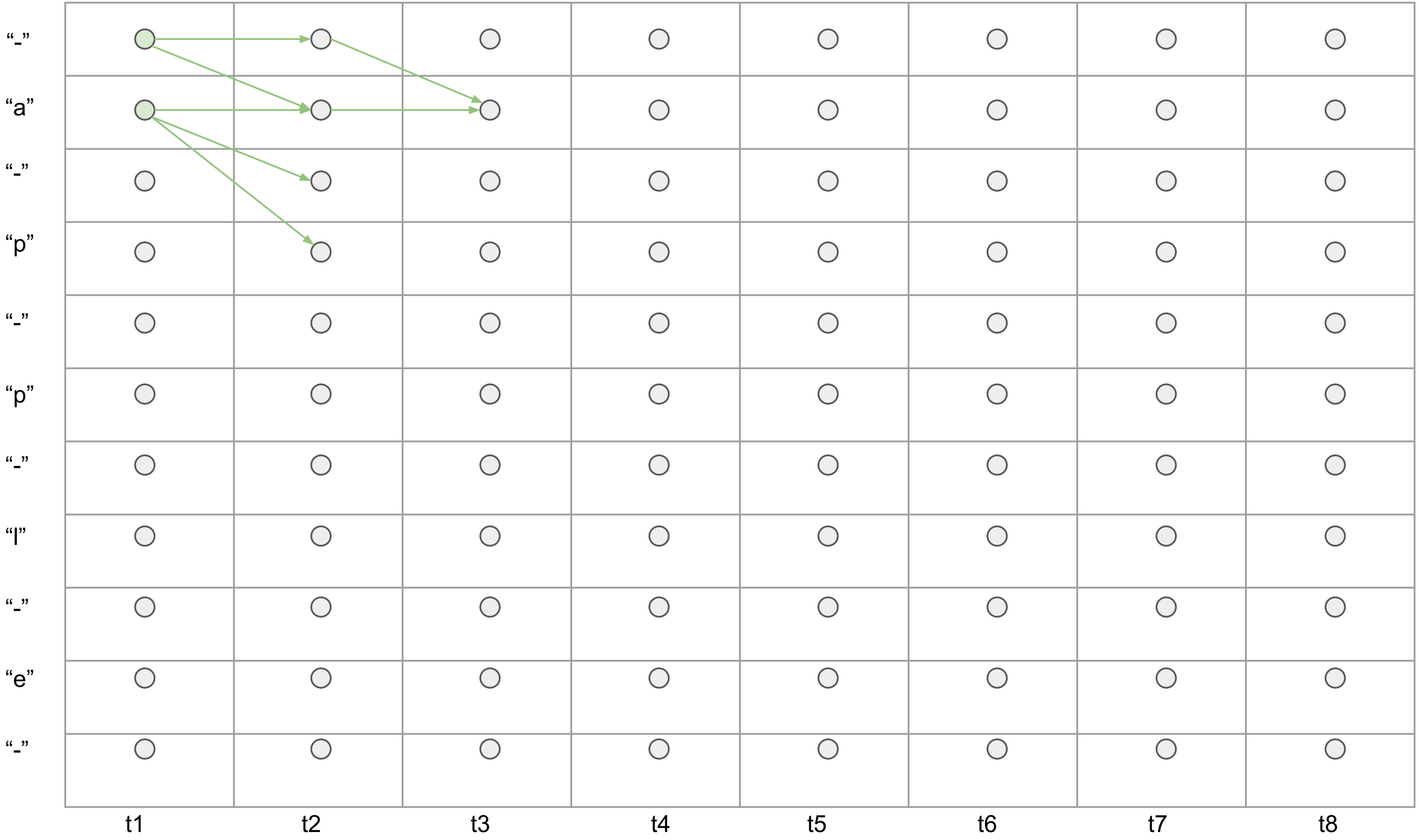


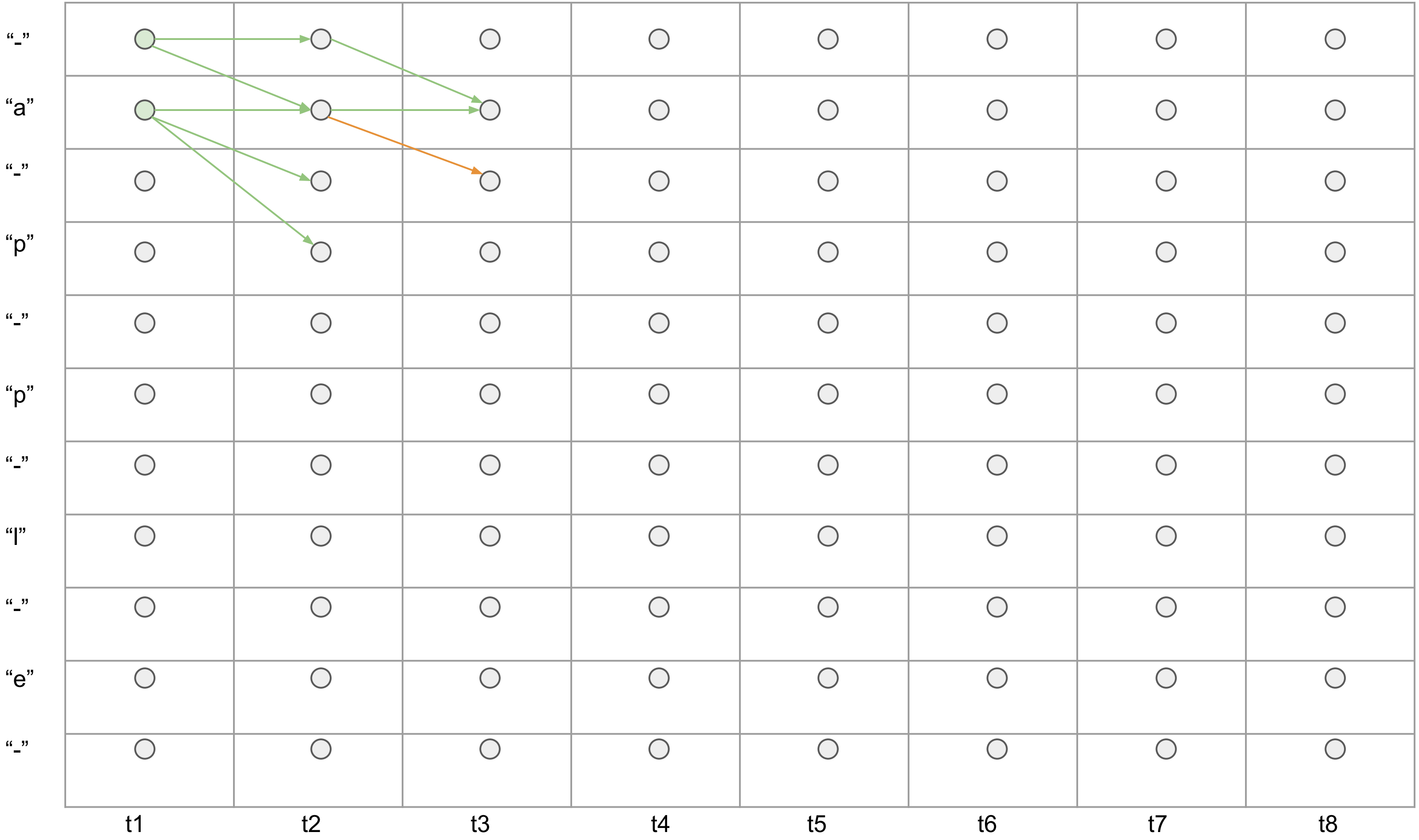
Impossible transitions



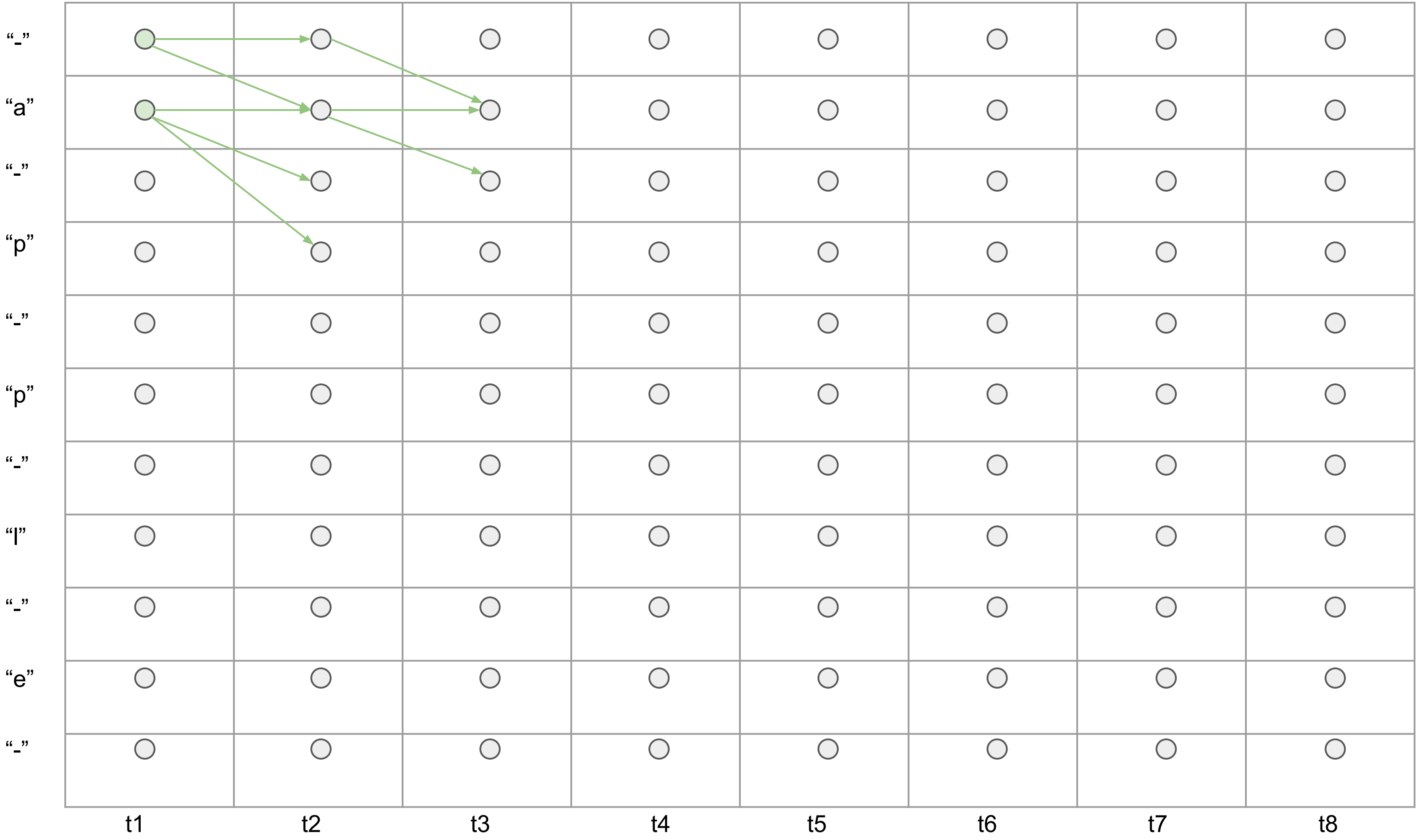

“-”
“a”
“-”
“p”
“-”
“p”
“-”
“l”
“-”
“e”
“-”
t1
t2
t3
t4
t5
t6
t7
t8

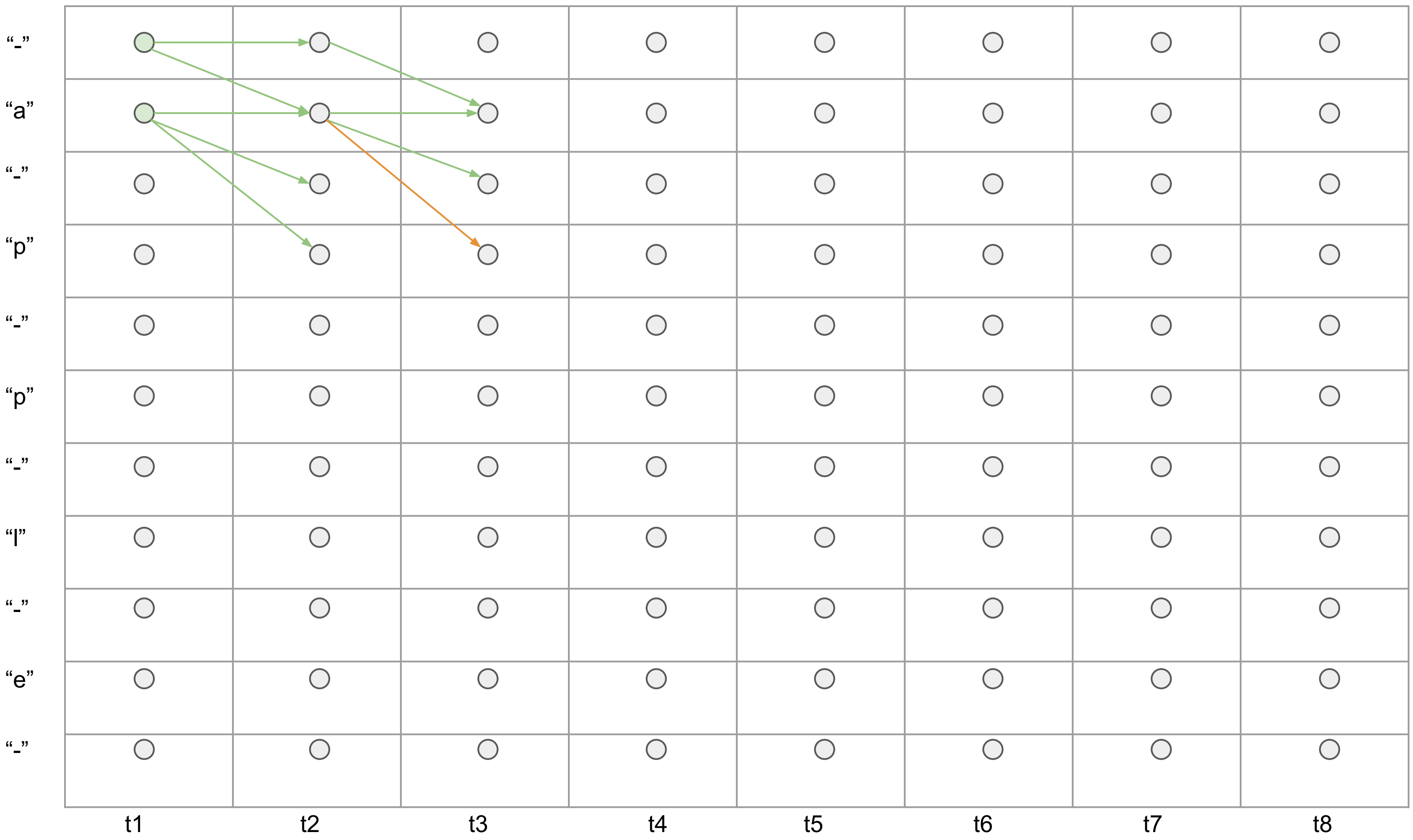
Possible transition, because paths can start with “aaa”. Example of valid path: “aaap-ple”. B(“aaap-ple”)=“apple”



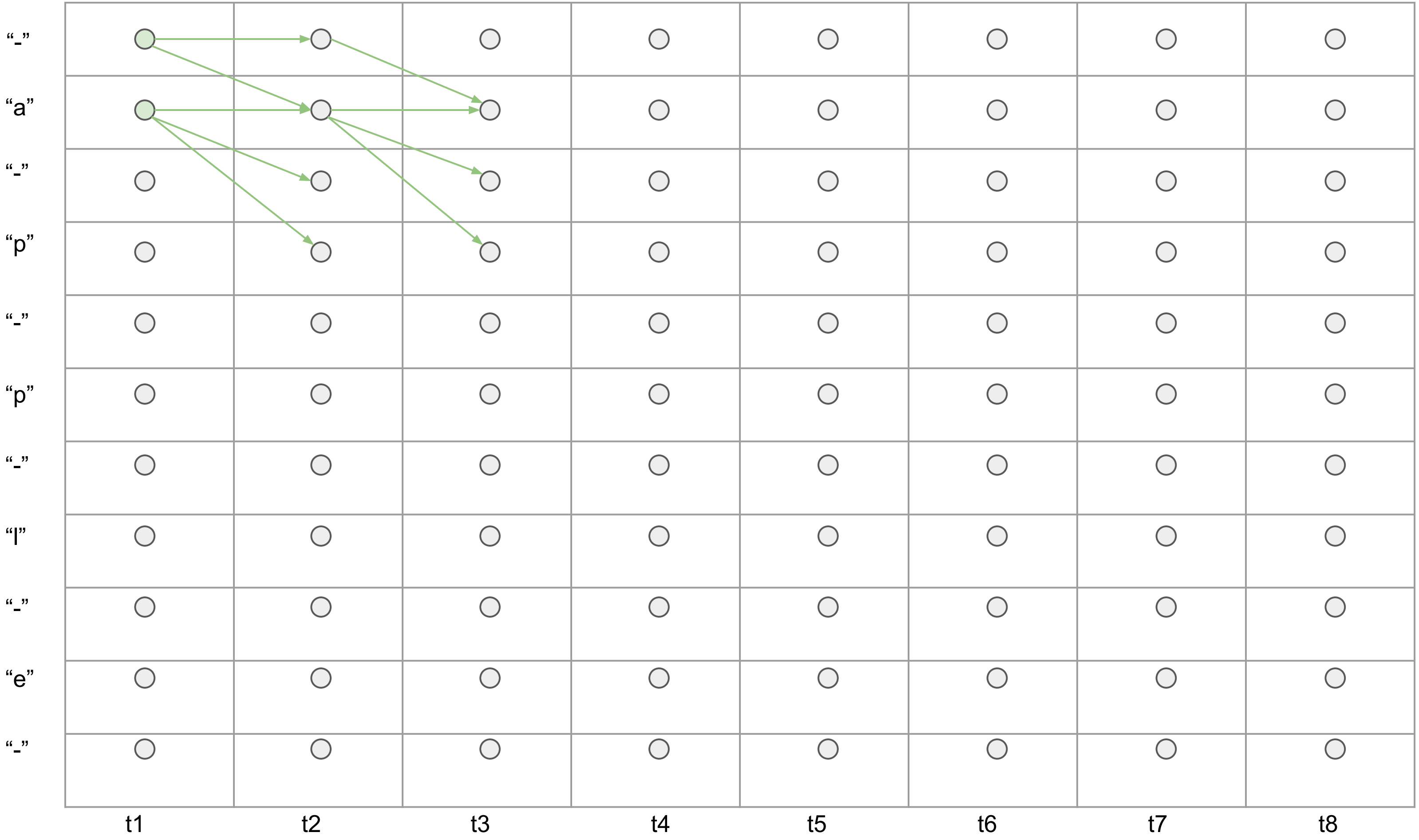

“-”
“a”
“-”
“p”
“-”
“p”
“-”
“l”
“-”
“e”
“-”
t1
t2
t3
t4
t5
t6
t7
t8

Possible transition, because paths can start with “aa-”. Example of valid path: “aa-p-ple”. B(“aa-p-ple”)=“apple”

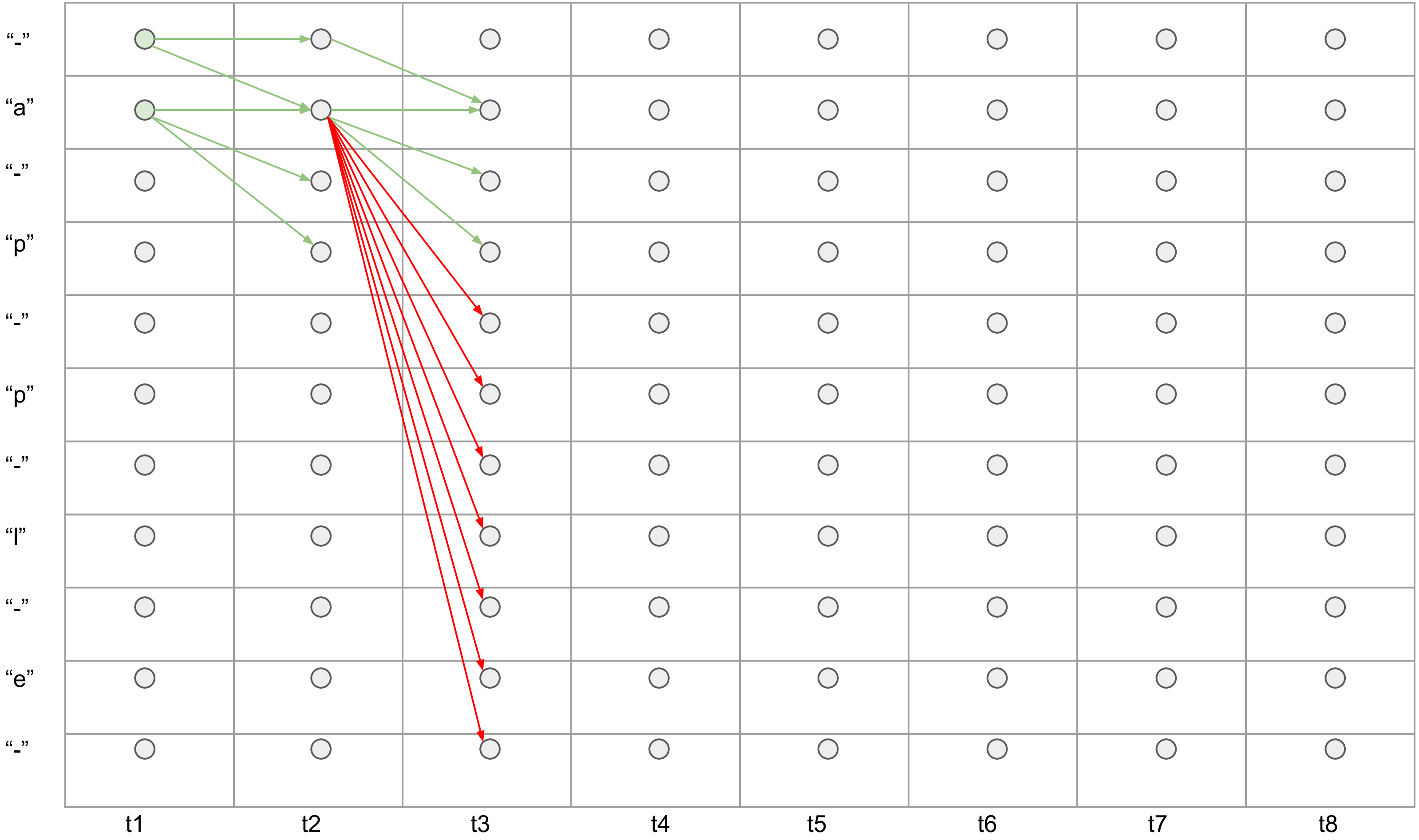



“-”
“a”
“-”
“p”
“-”
“p”
“-”
“l”
“-”
“e”
“-”
t1
t2
t3
t4
t5
t6
t7
t8

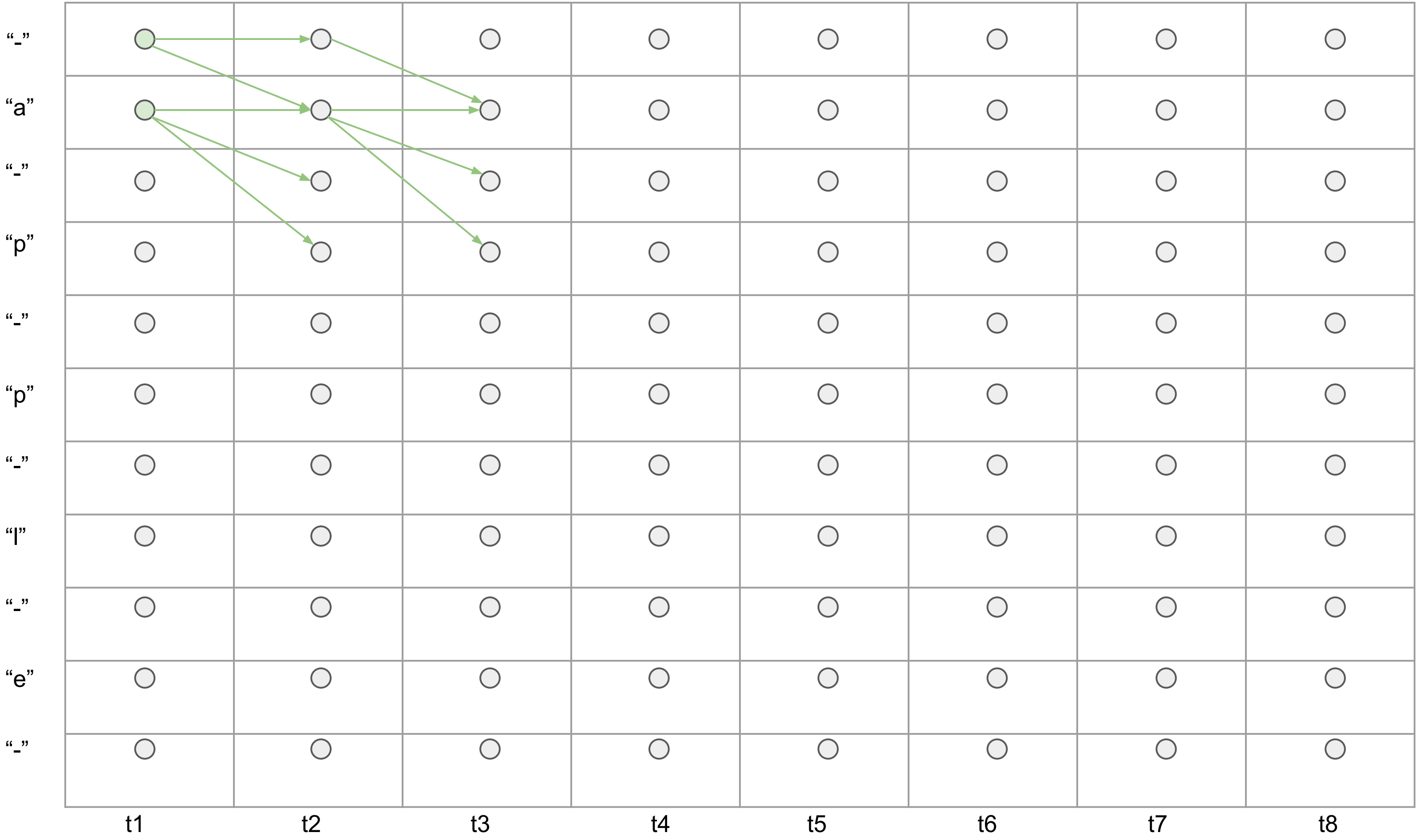
Possible transition, because paths can start with “aap”. Example of valid path: “aapp-ple”. B(“aapp-ple”)=“apple”



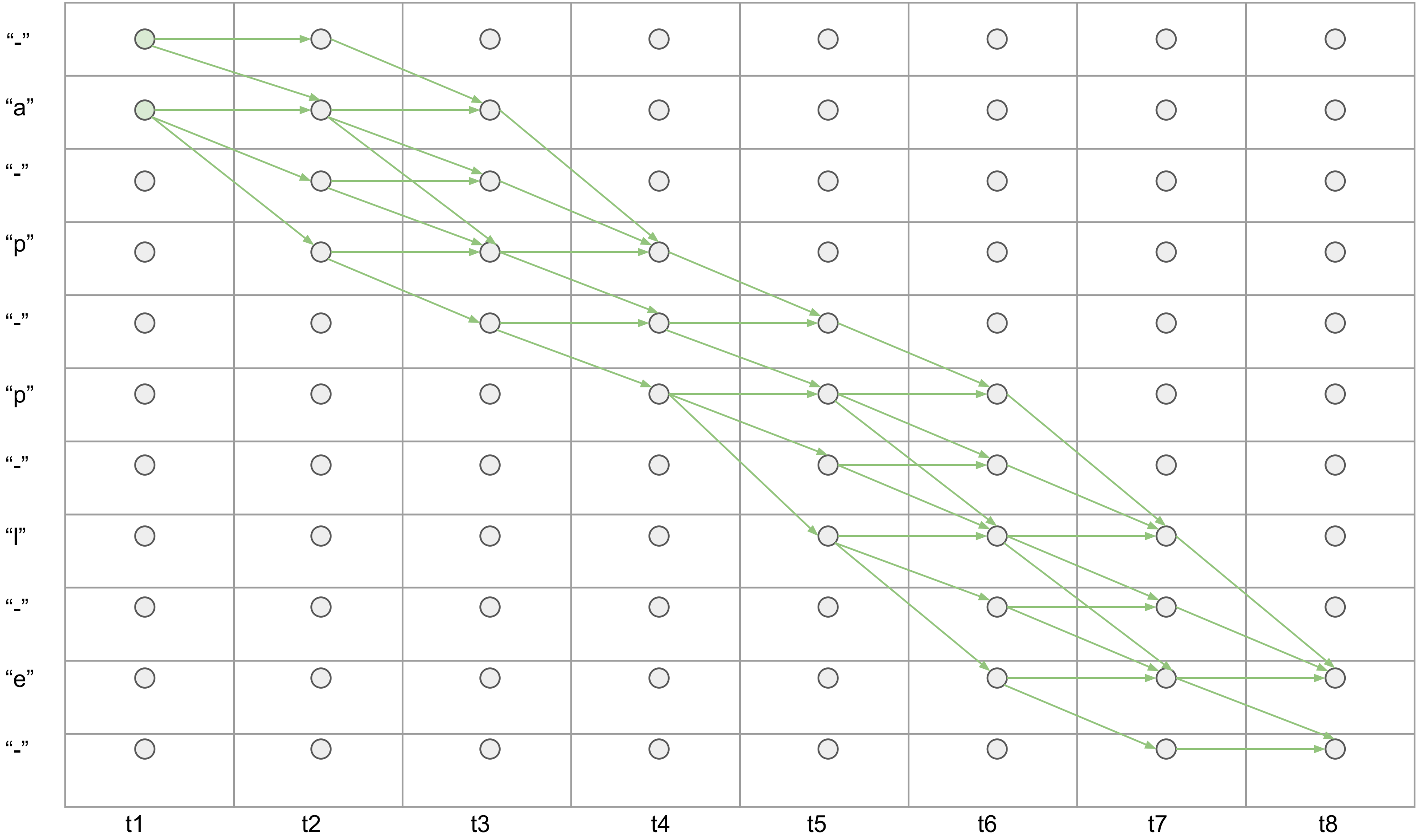
Impossible transitions



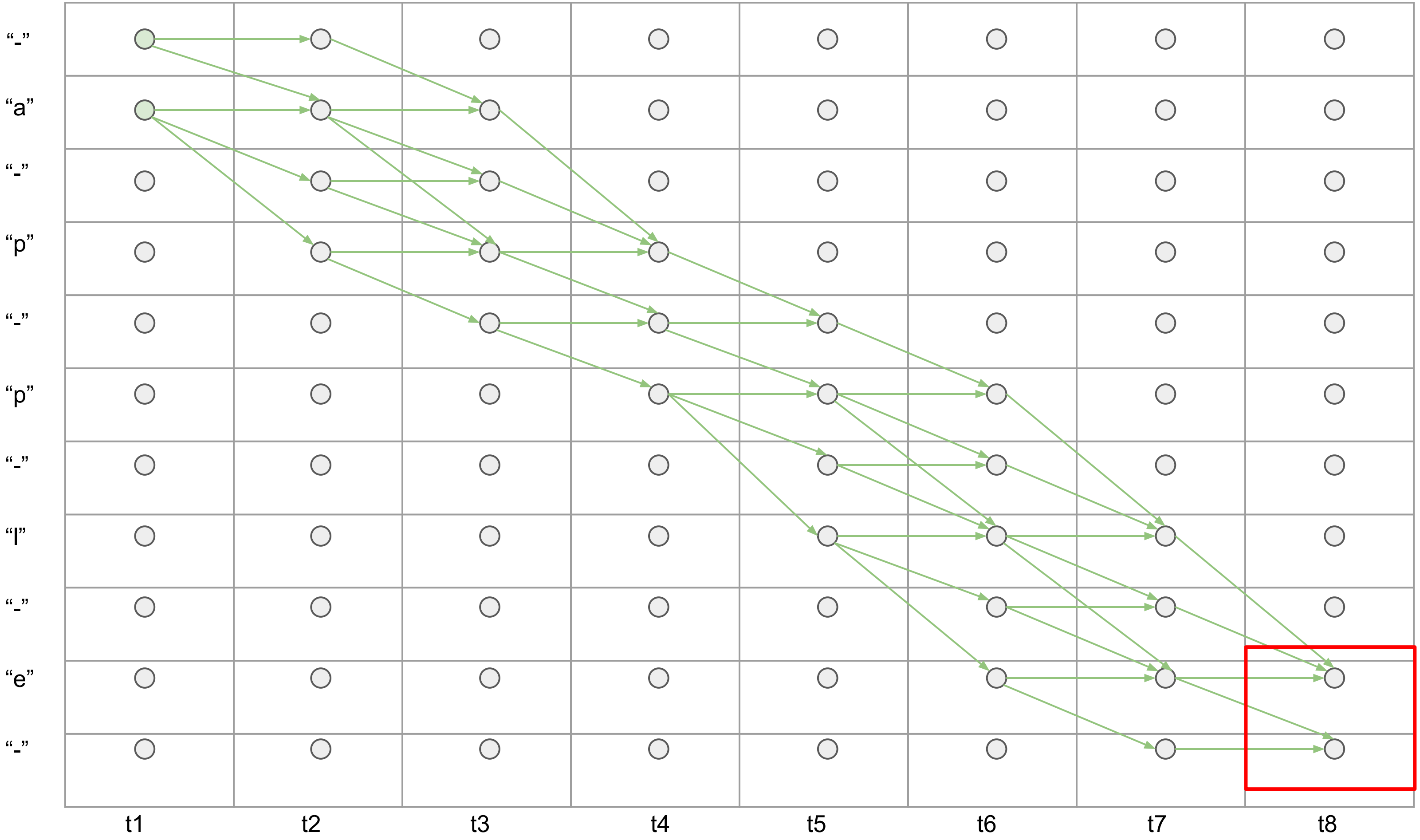
We can continue to add valid transactions by the same logic.



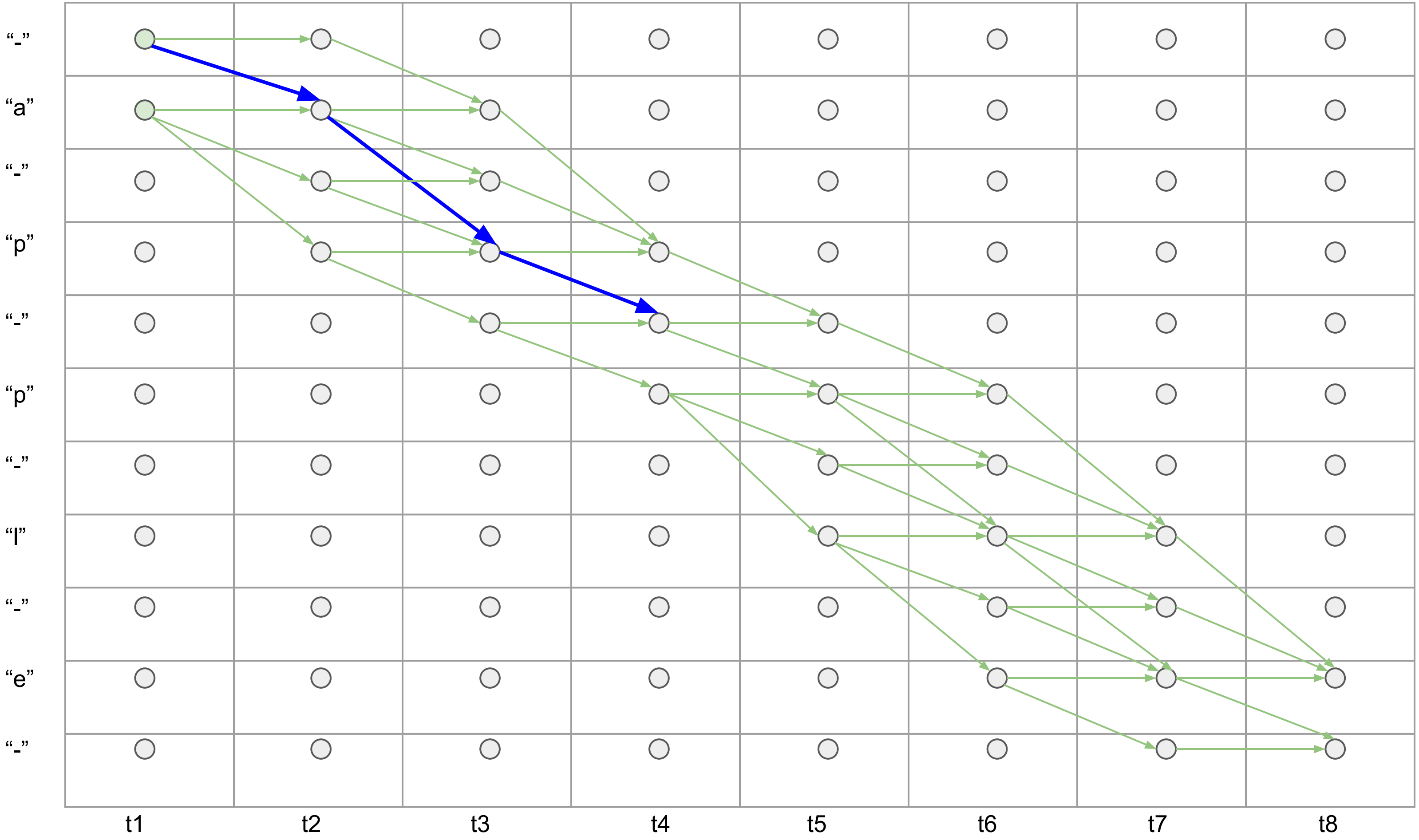
We can continue to add valid transactions by the same logic. Here is the result.



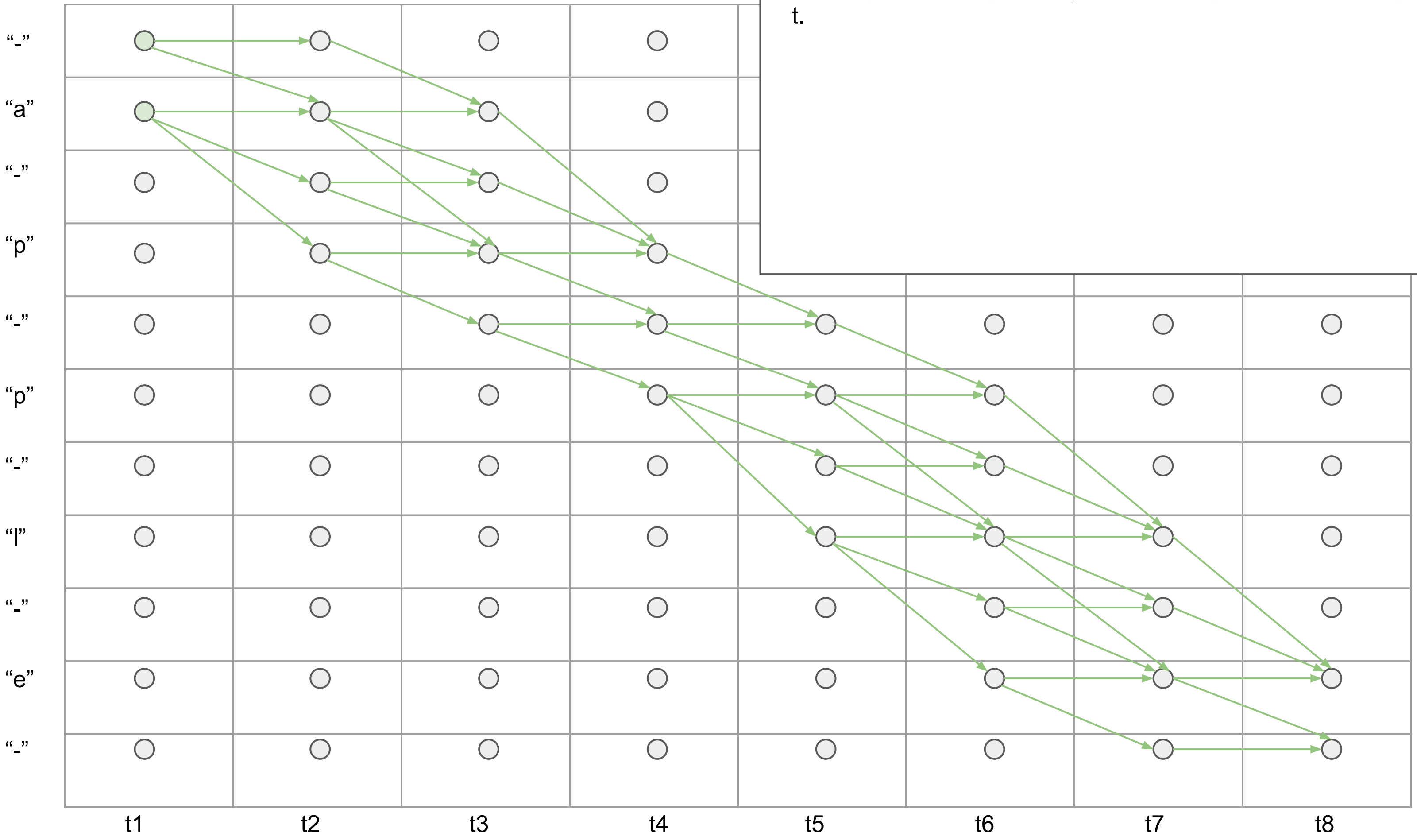
Interesting note: all valid paths should end up with this nodes.



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 - $p(“-ap-”) = y^1_- \cdot y^2_a \cdot y^3_p \cdot y^4_-$

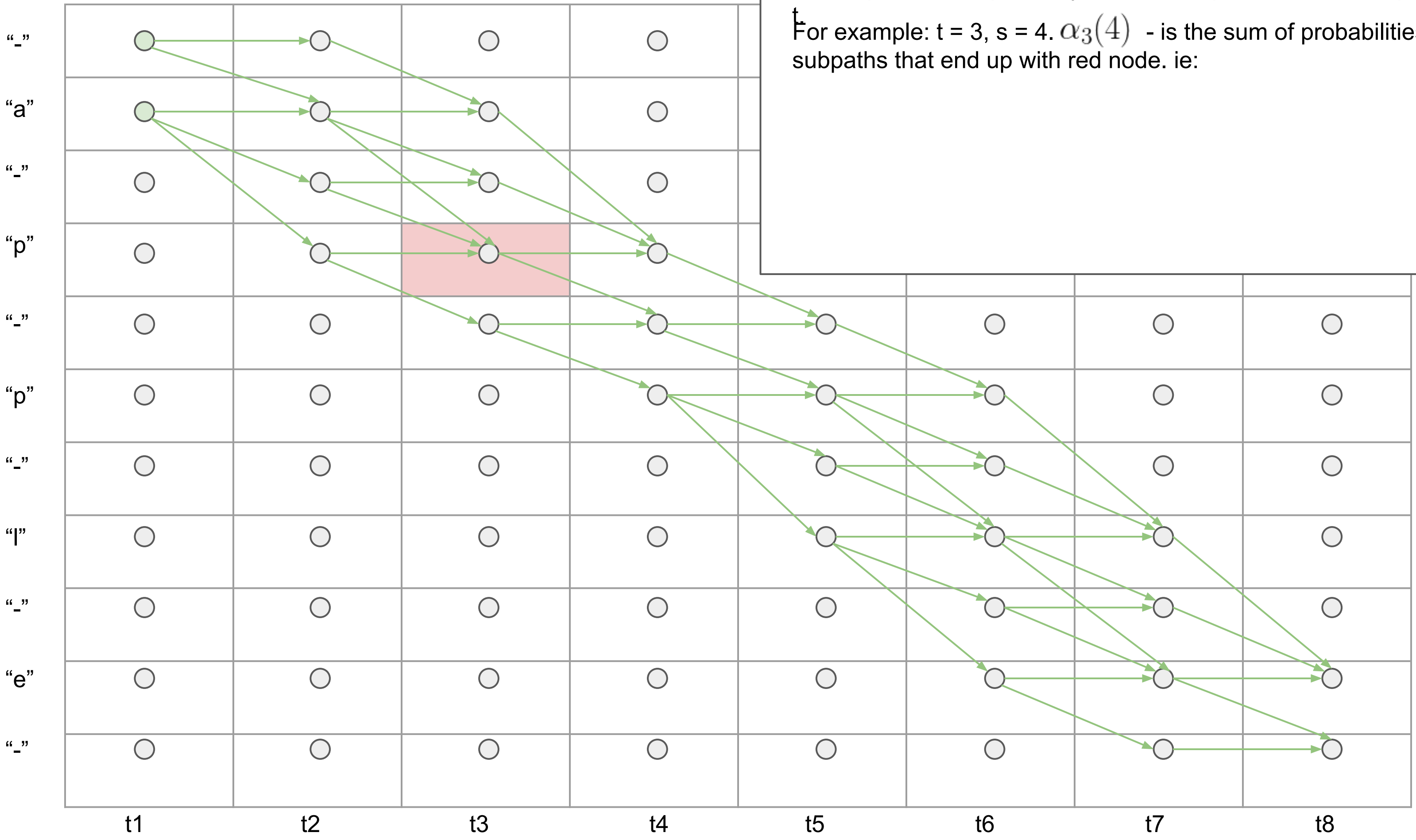


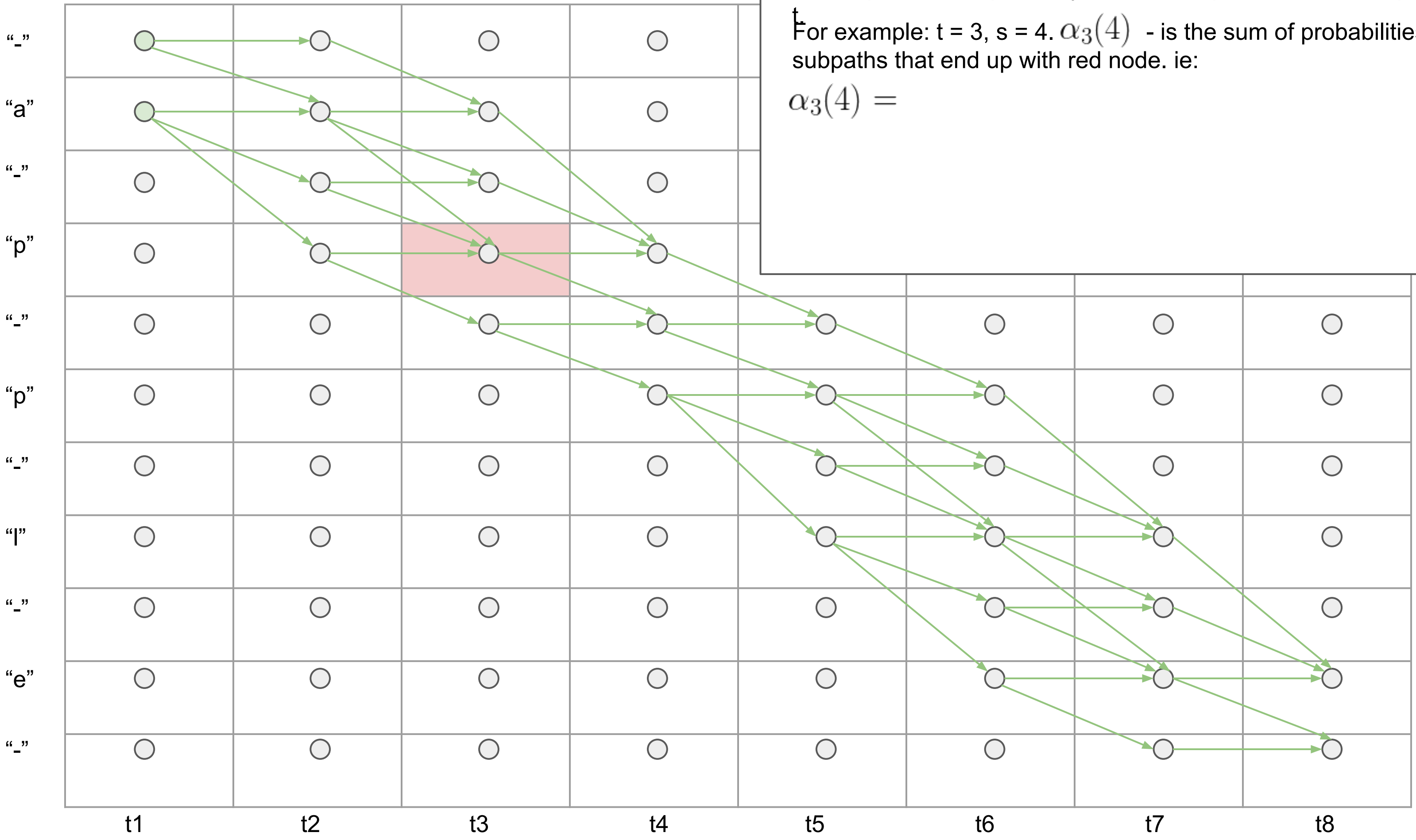
Let me define variable $\alpha_t(s)$ - the total probability of all subpaths, whose prefix end up with symbol at s-th position in the sequence at time t.



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:





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.

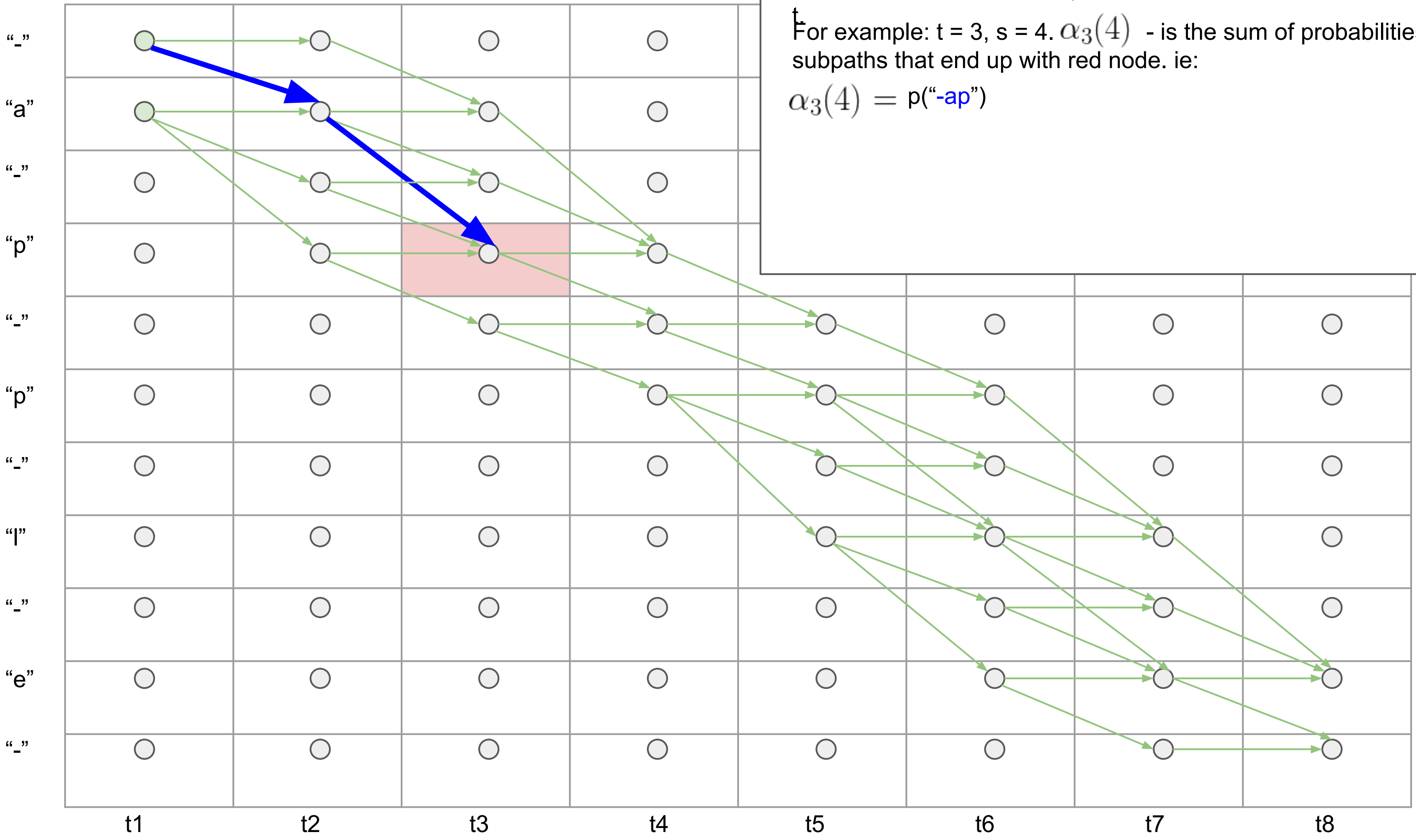
For example: t = 3, s = 4. $\alpha_3(4)$ - is the sum of probabilities of all subpaths that end up with red node. ie:

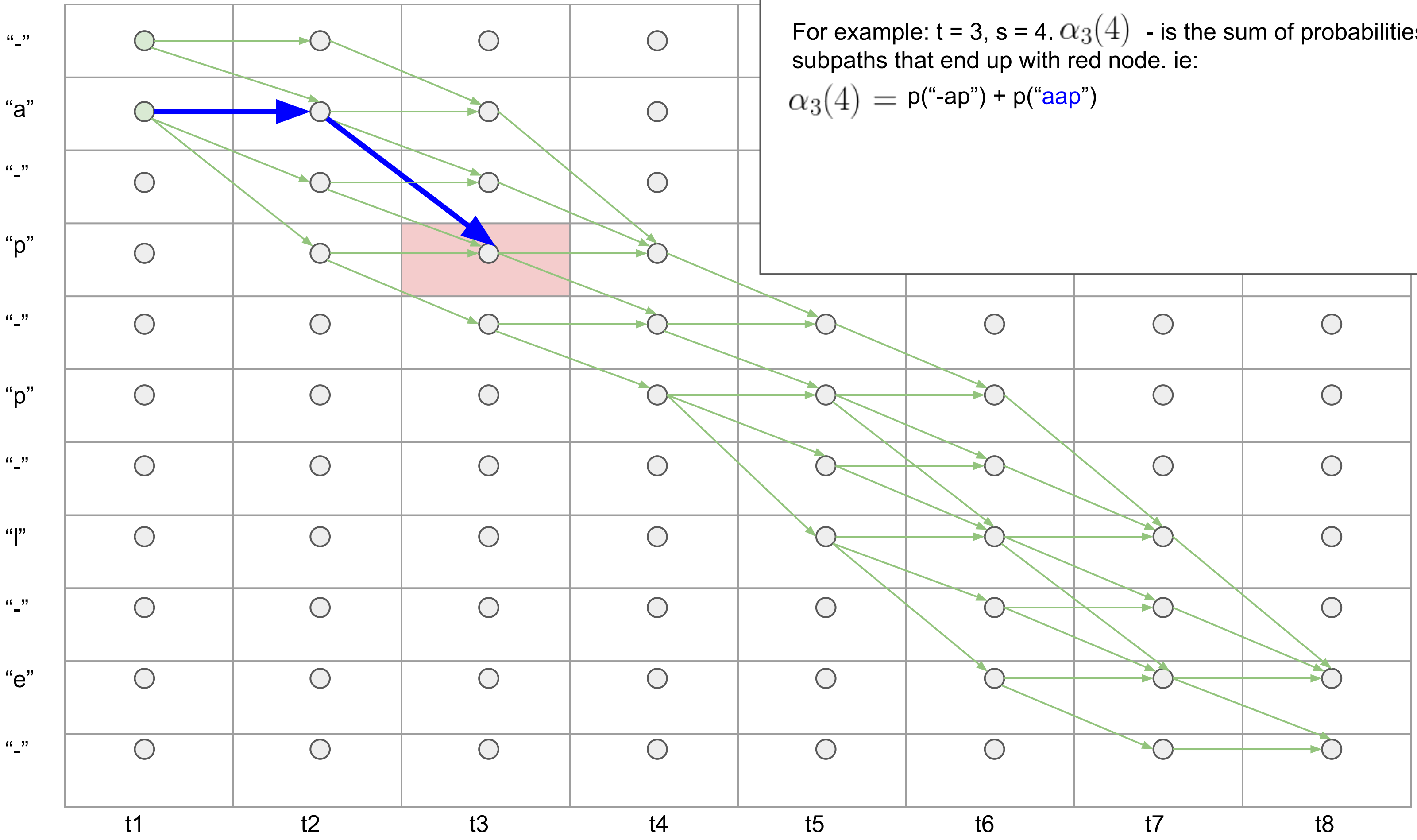
$$\alpha_3(4) =$$

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”)

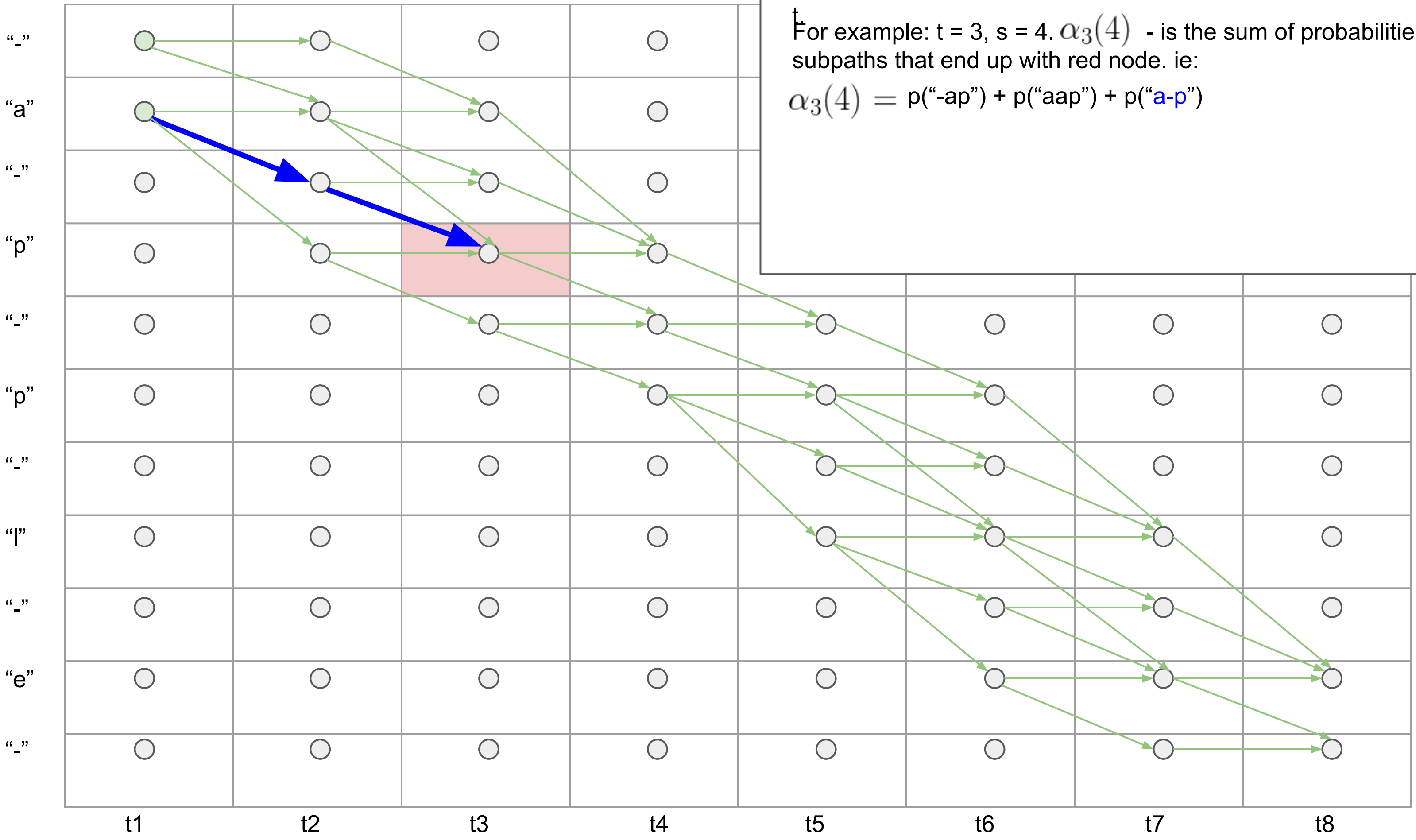




Let me define variable $\alpha_t(s)$ - the total probability of all subpaths, that end up with symbol at s-th position in the sequence at time t.

For example: t = 3, s = 4. $\alpha_3(4)$ - is the sum of probabilities of all subpaths that end up with red node. ie:

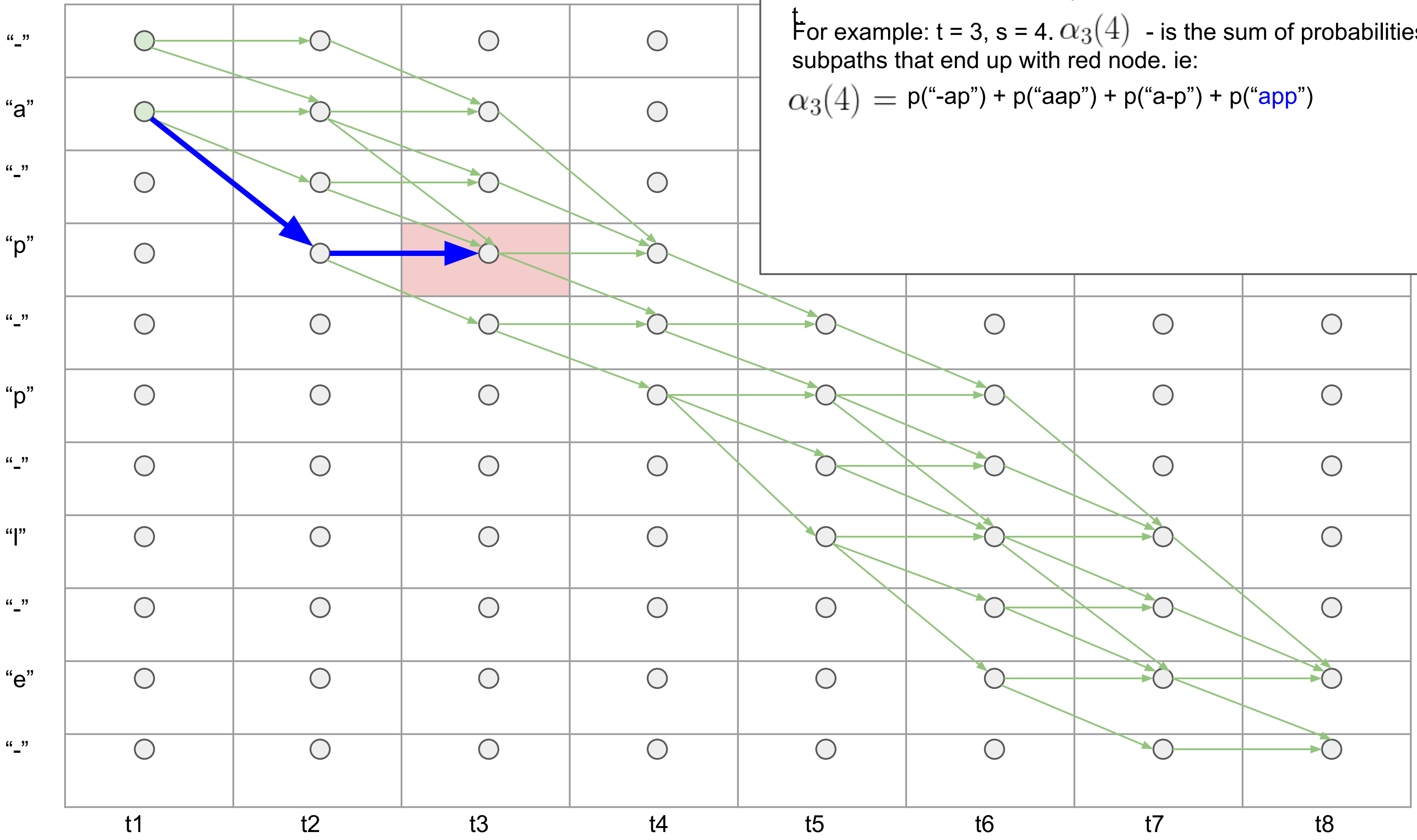
$\alpha_3(4) =$ p(“-ap”) + p(“aap”)



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")



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.

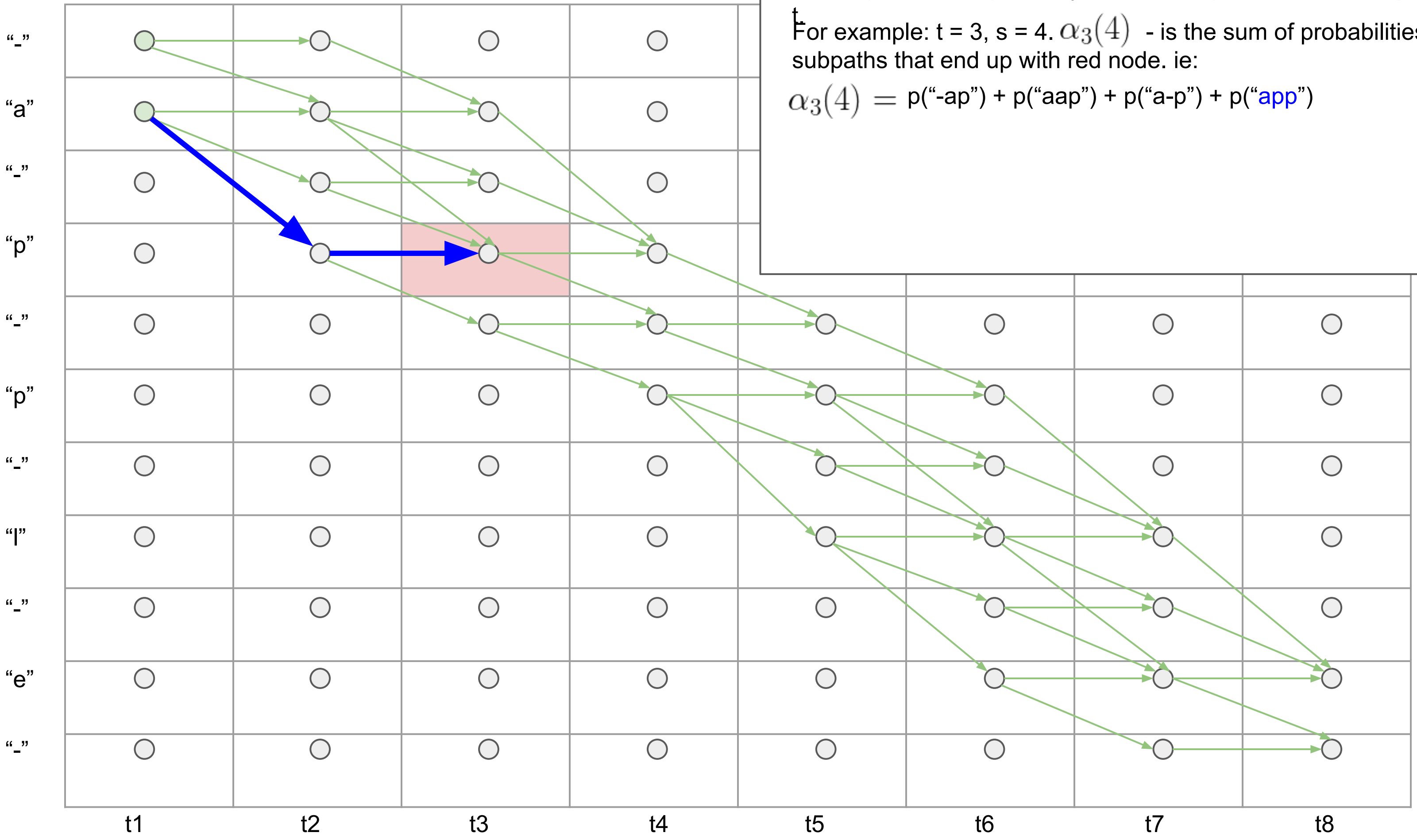
For example: t = 3, s = 4. $\alpha_3(4)$ - is the sum of probabilities of all subpaths that end up with red node. ie:

$\alpha_3(4) =$ p(“-ap”) + p(“aap”) + p(“a-p”) + p(“app”)

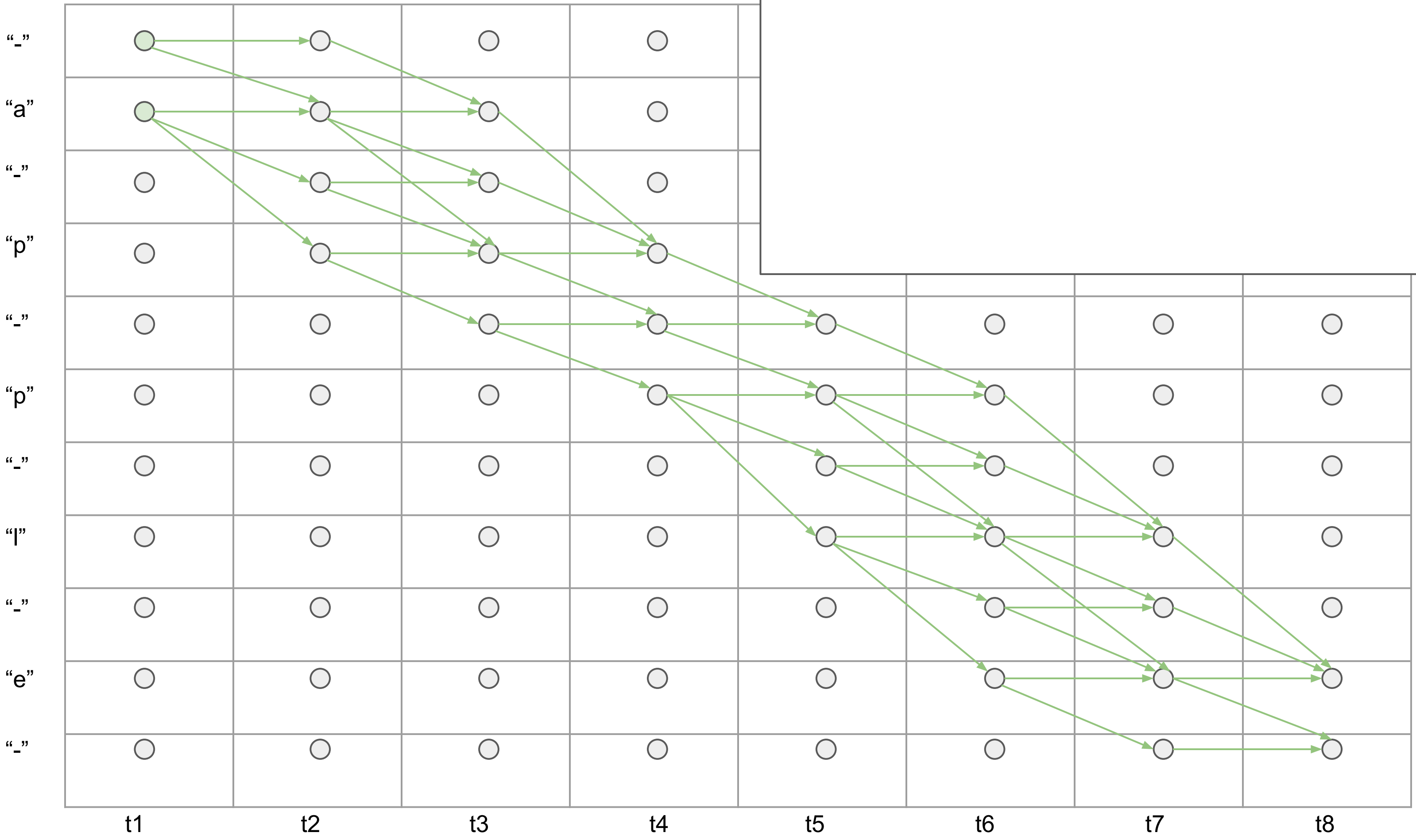
Let me define variable $\alpha_t(s)$ - the total probability of all subpaths, whose prefix end up with symbol at s-th position in the sequence at time t.

For example: t = 3, s = 4. $\alpha_3(4)$ - is the sum of probabilities of all subpaths that end up with red node. ie:

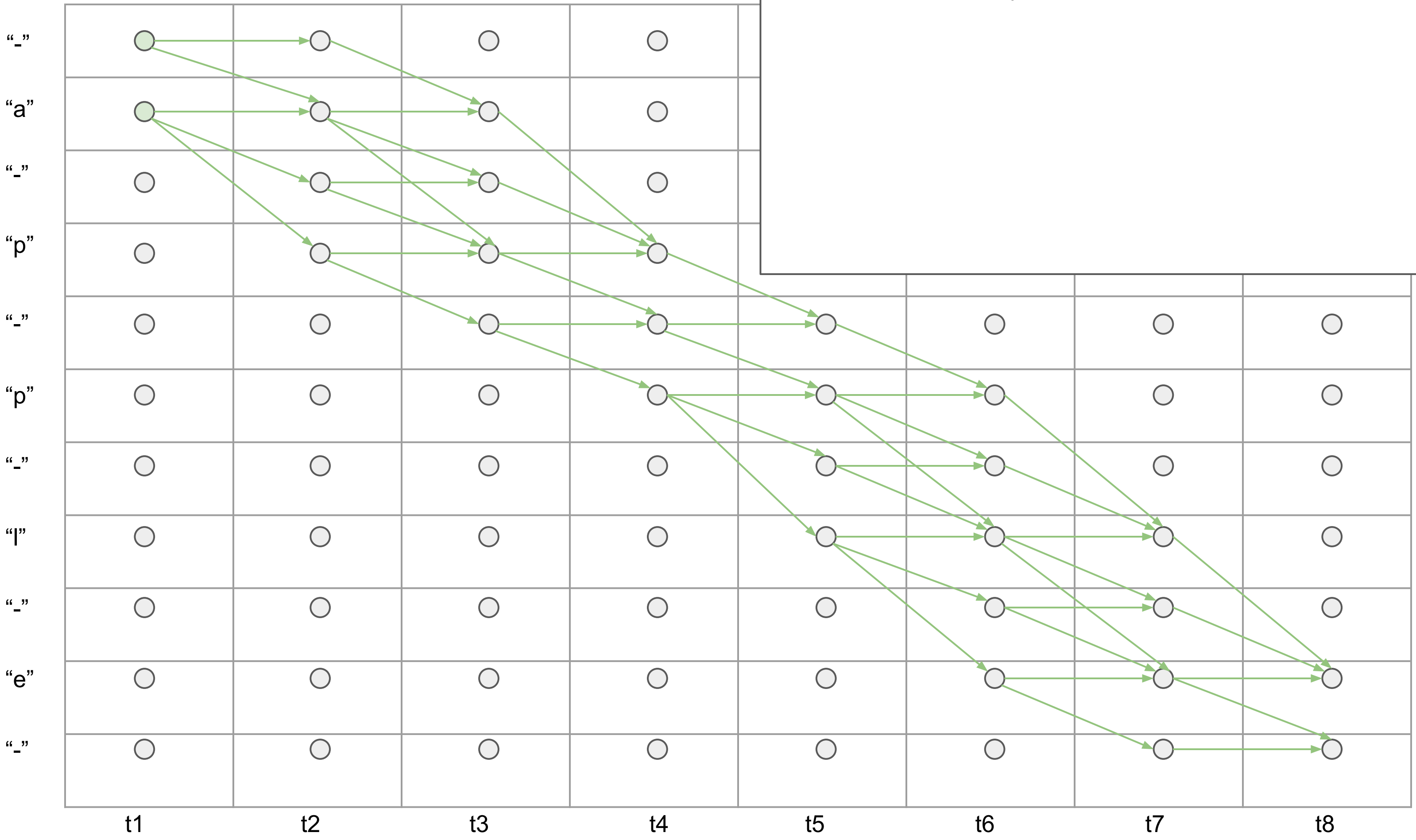
$\alpha_3(4) =$ p(“-ap”) + p(“aap”) + p(“a-p”) + p(“app”)



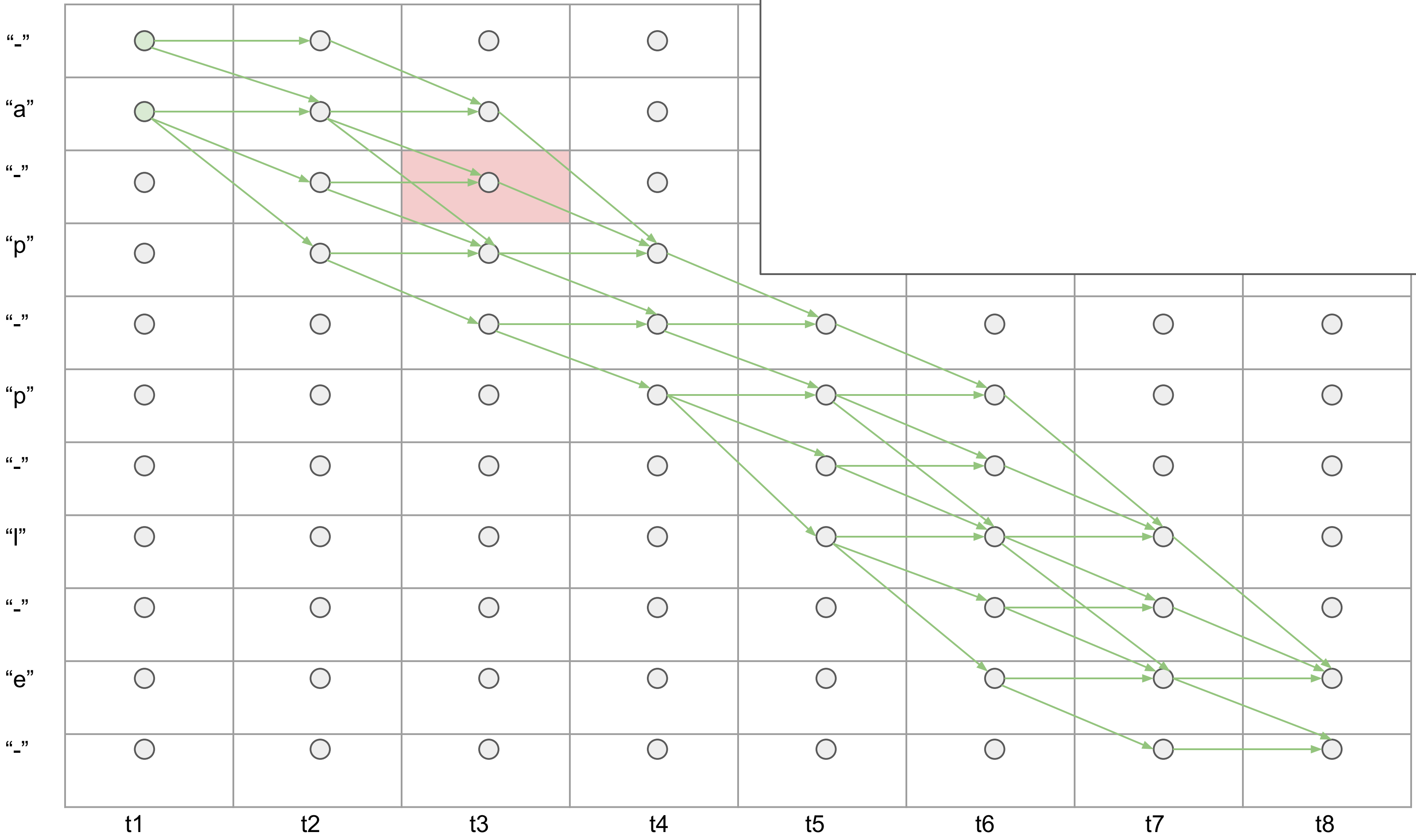
For each cell of this table we can compute $\alpha_t(s)$.



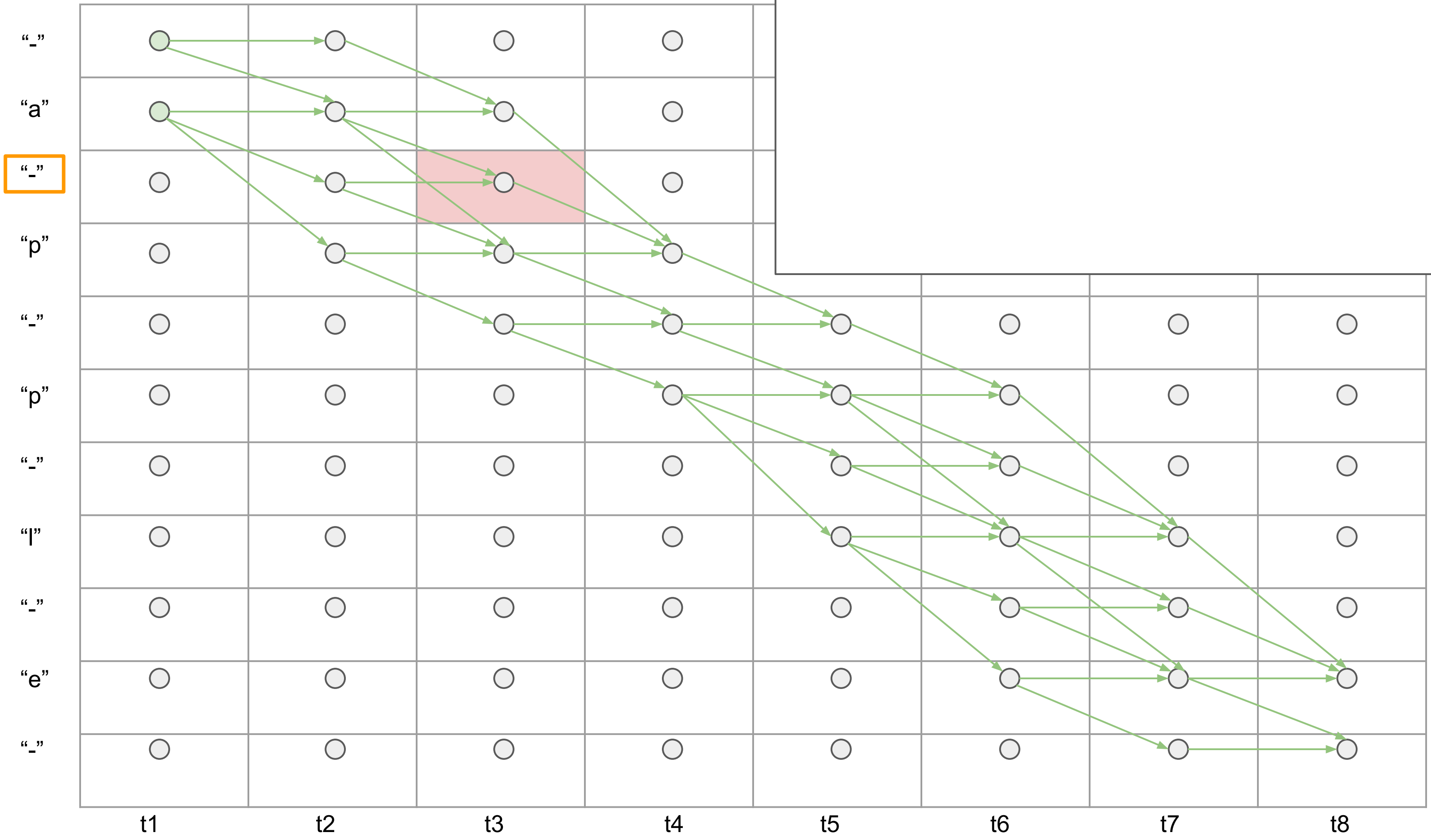
For each cell of this table we can compute $\alpha_t(s)$. We also can calculate it recurrently. There are three cases, that we should review.



Case1. s-th symbol is blank. Example: s=3, t=3; 3-rd symbol is “-”

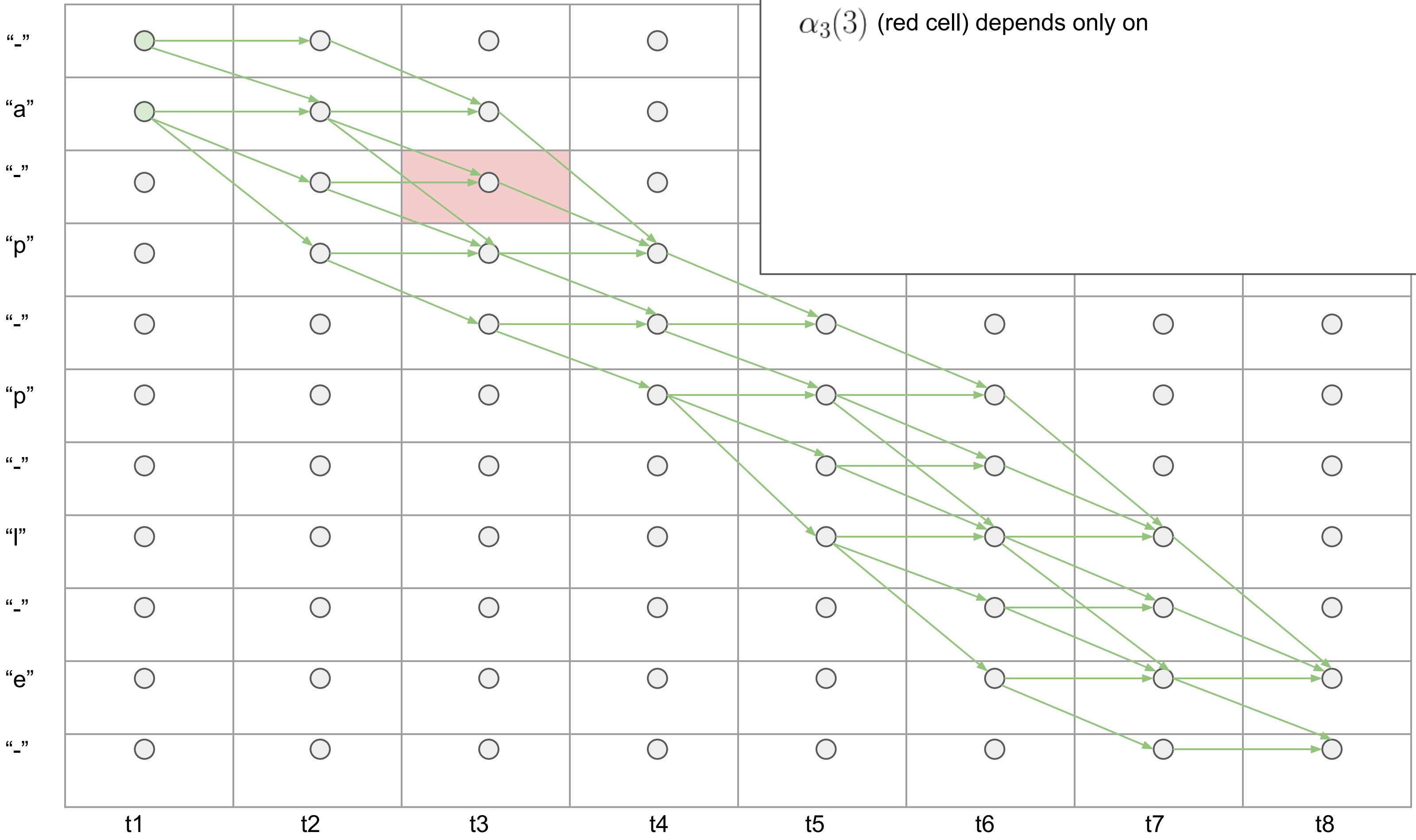


Case1. s-th symbol is blank. Example: s=3, t=3; 3-rd symbol is “-’



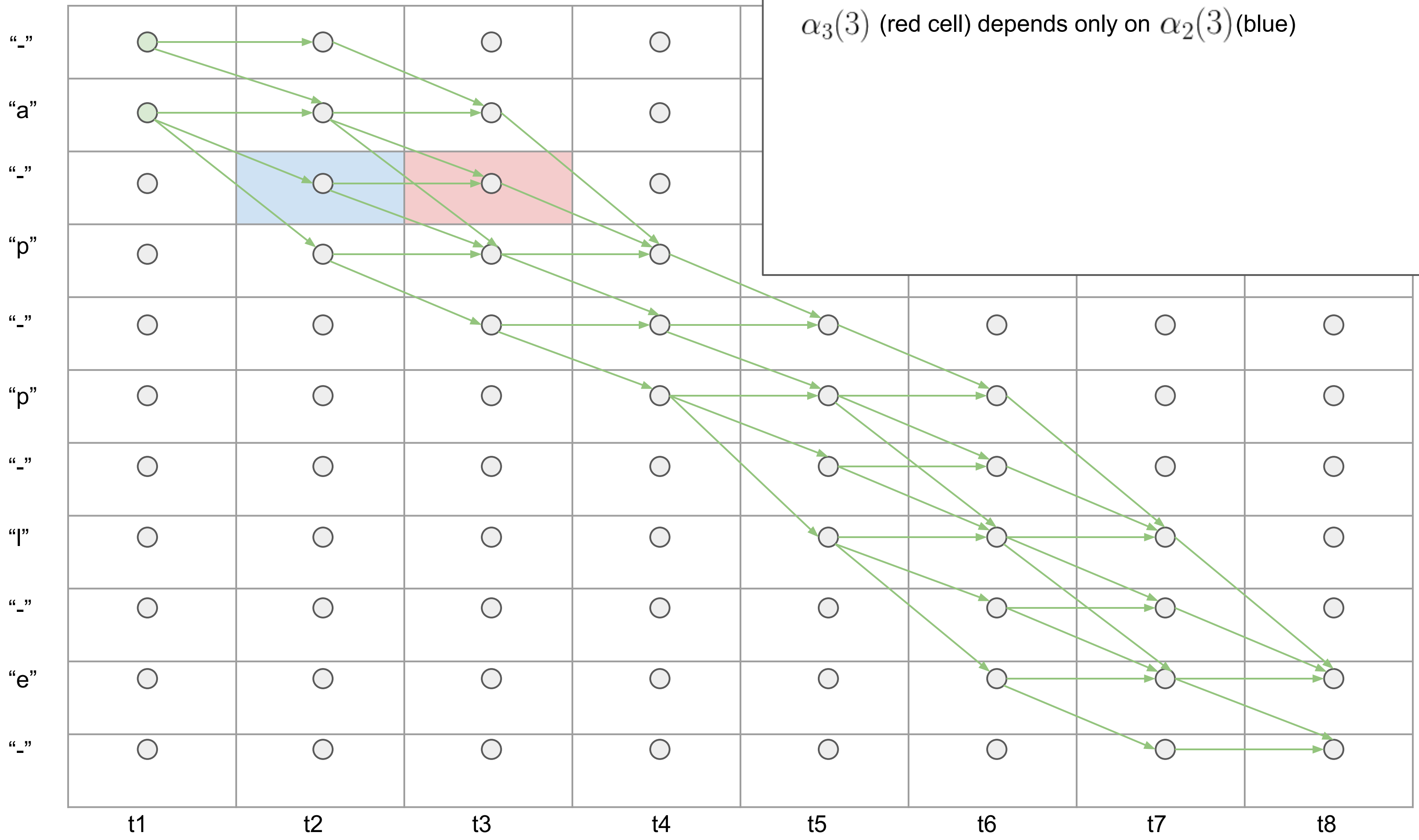
Case1. s-th symbol is blank. Example: s=3, 3-rd symbol is “-”

$\alpha_3(3)$ (red cell) depends only on



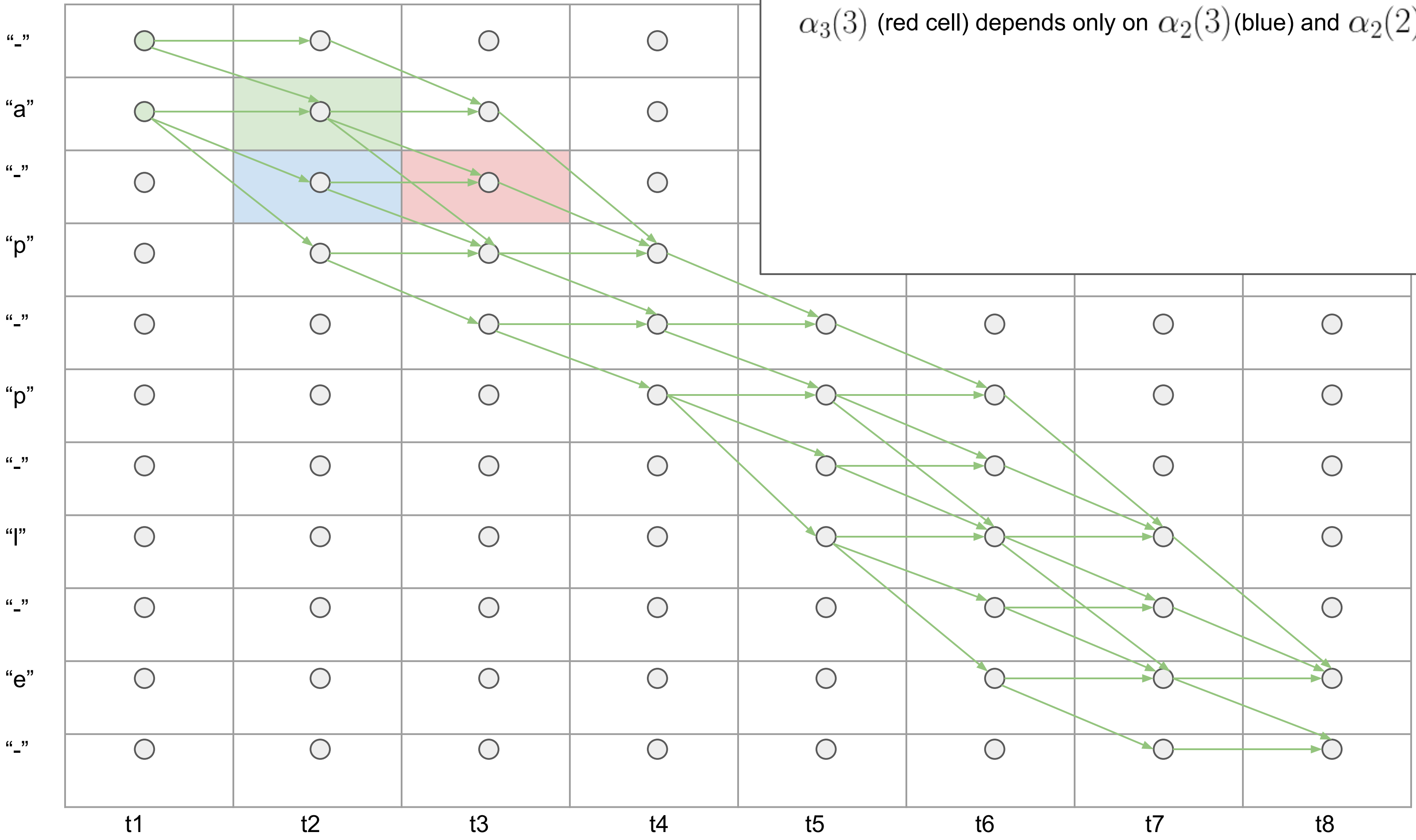
Case1. s-th symbol is blank. Example: s=3, 3-rd symbol is “-”

$\alpha_3(3)$ (red cell) depends only on $\alpha_2(3)$(blue)



Case1. s-th symbol is blank. Example: s=3, 3-rd symbol is “-”

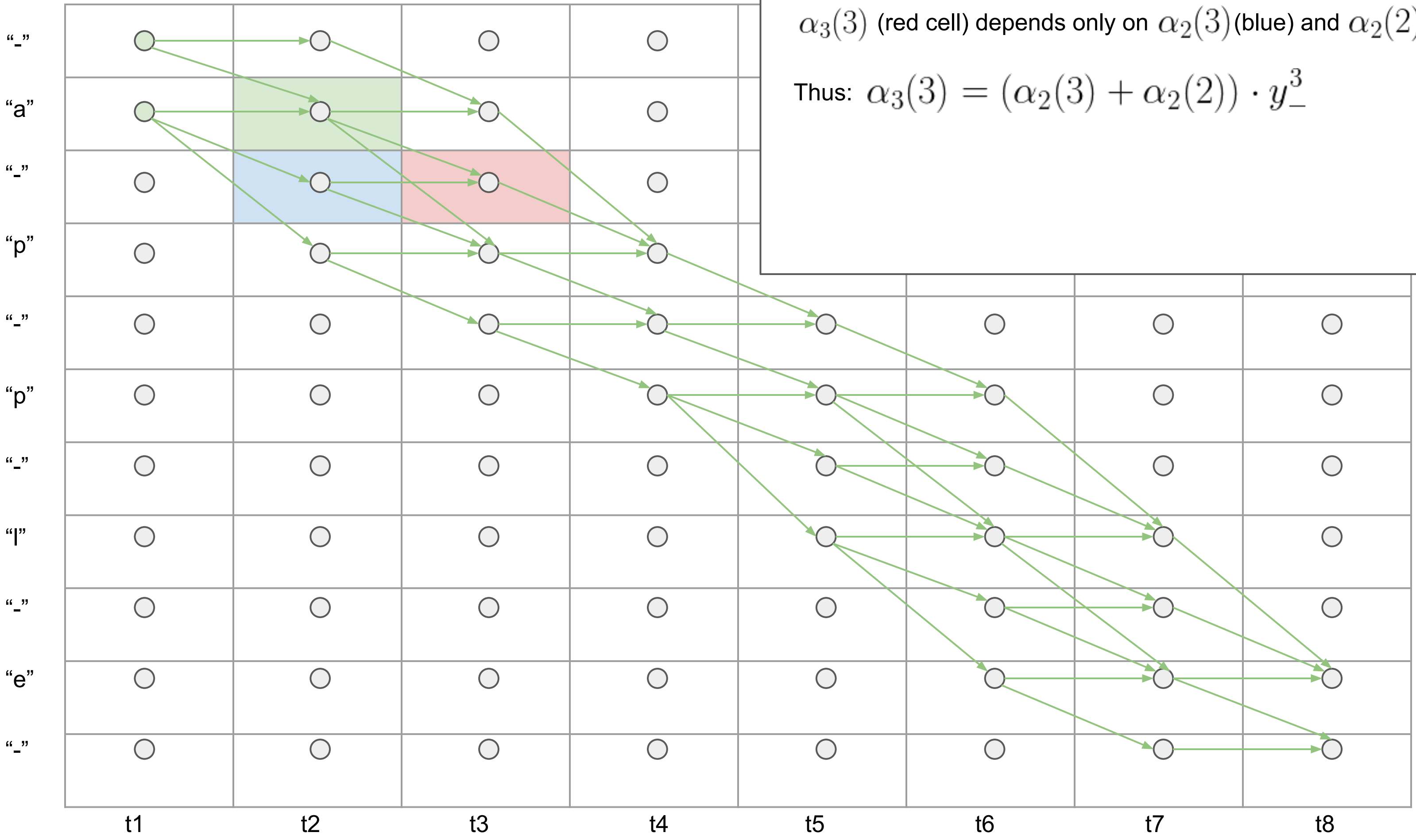
$\alpha_3(3)$ (red cell) depends only on $\alpha_2(3)$(blue) and $\alpha_2(2)$(green)



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: $\alpha_3(3) = (\alpha_2(3) + \alpha_2(2)) \cdot y^3_-$

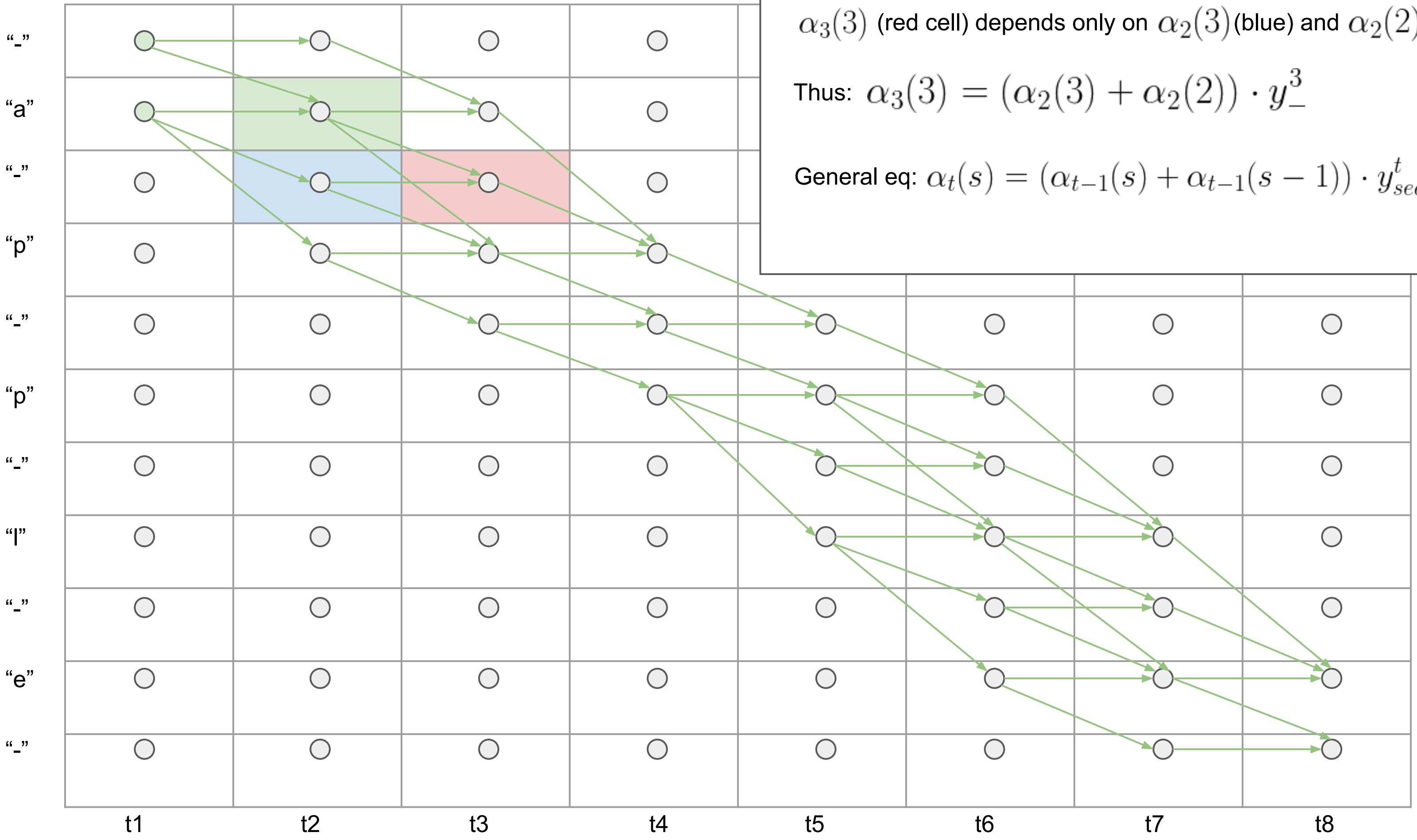


Case1. s-th symbol is blank. Example: s=3, 3-rd symbol is “-”

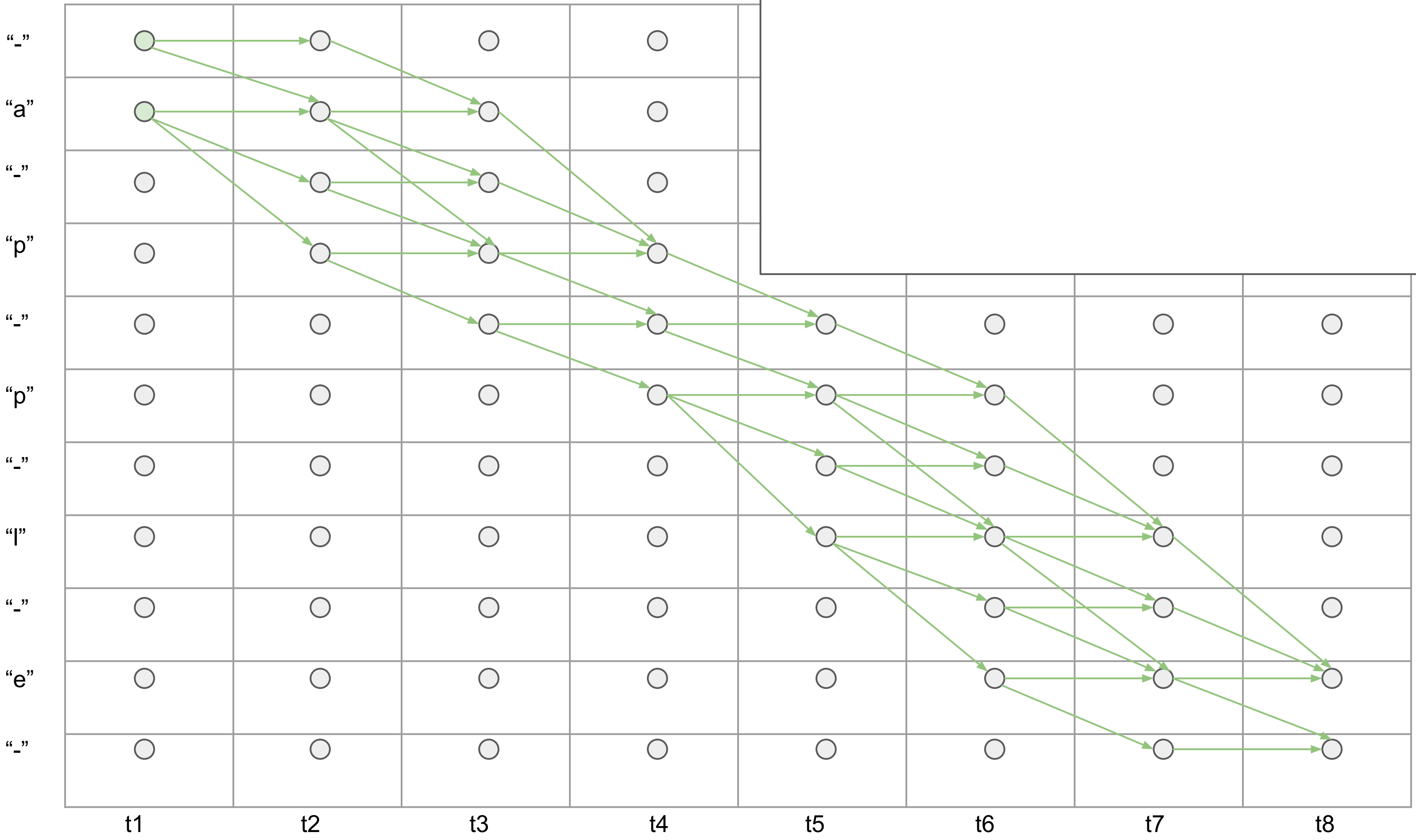
$\alpha_3(3)$ (red cell) depends only on $\alpha_2(3)$(blue) and $\alpha_2(2)$(green)

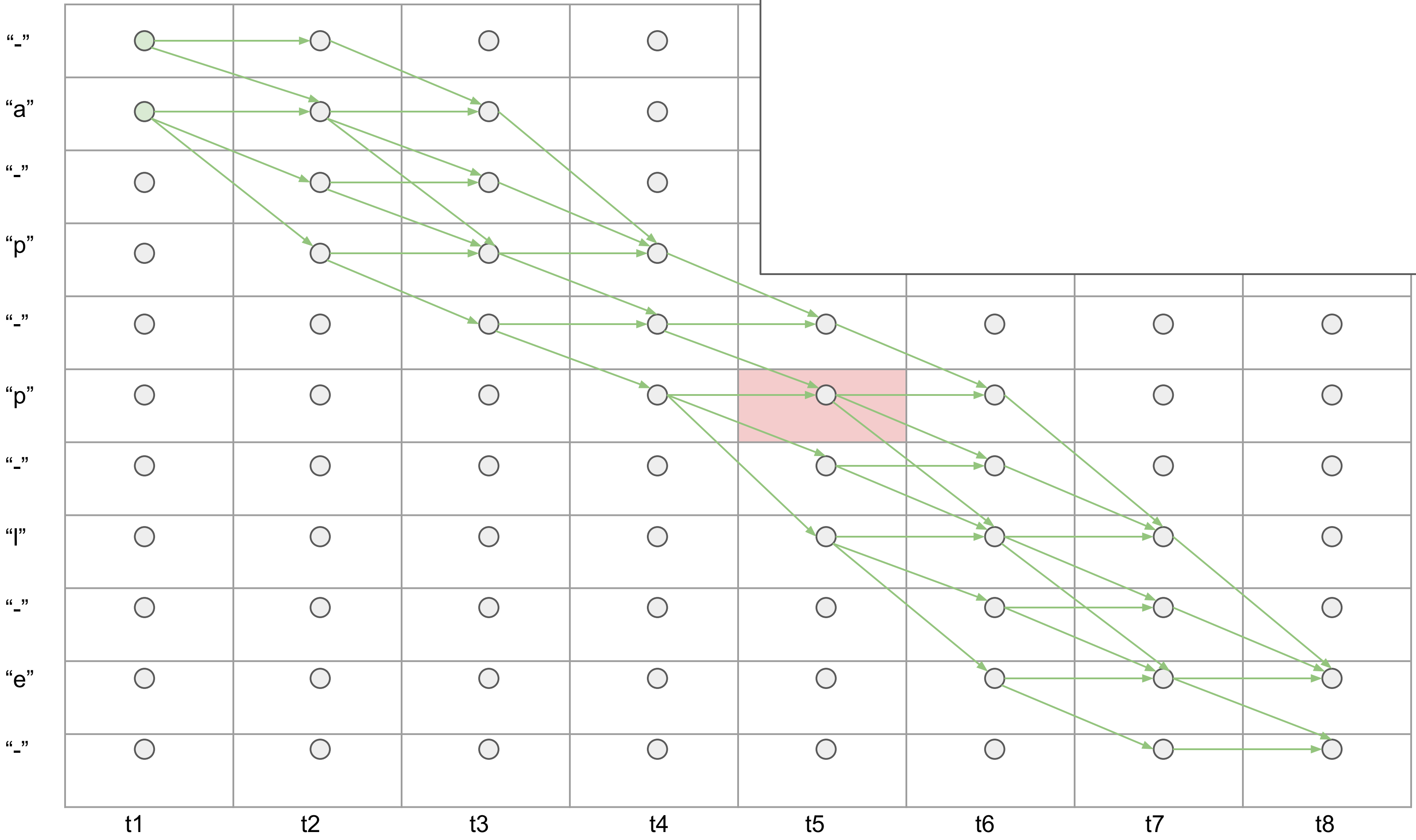
Thus: $\alpha_3(3) = (\alpha_2(3) + \alpha_2(2)) \cdot y^3_-$

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



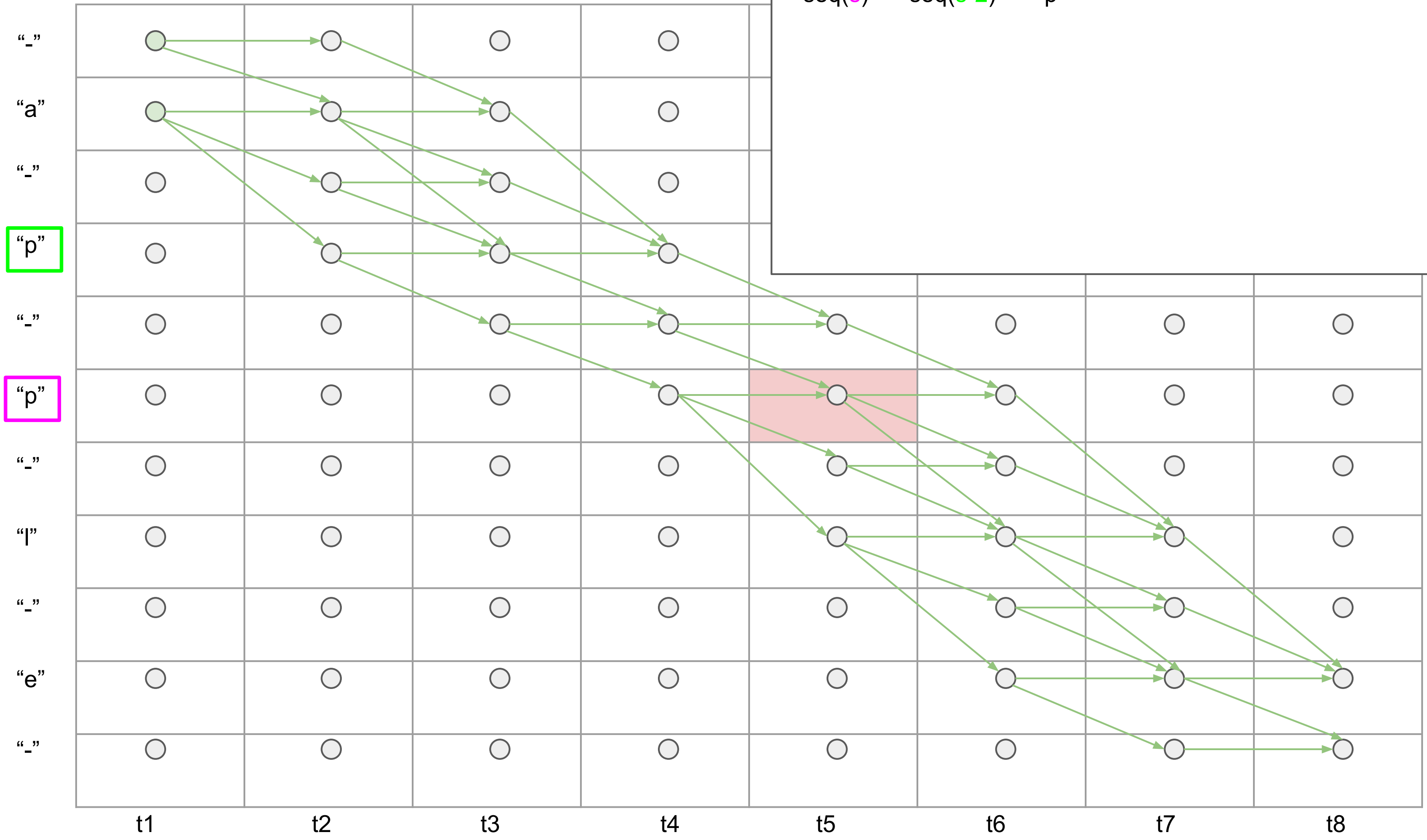
Case2. s-th symbol is equal to (s-2)-th symbol.





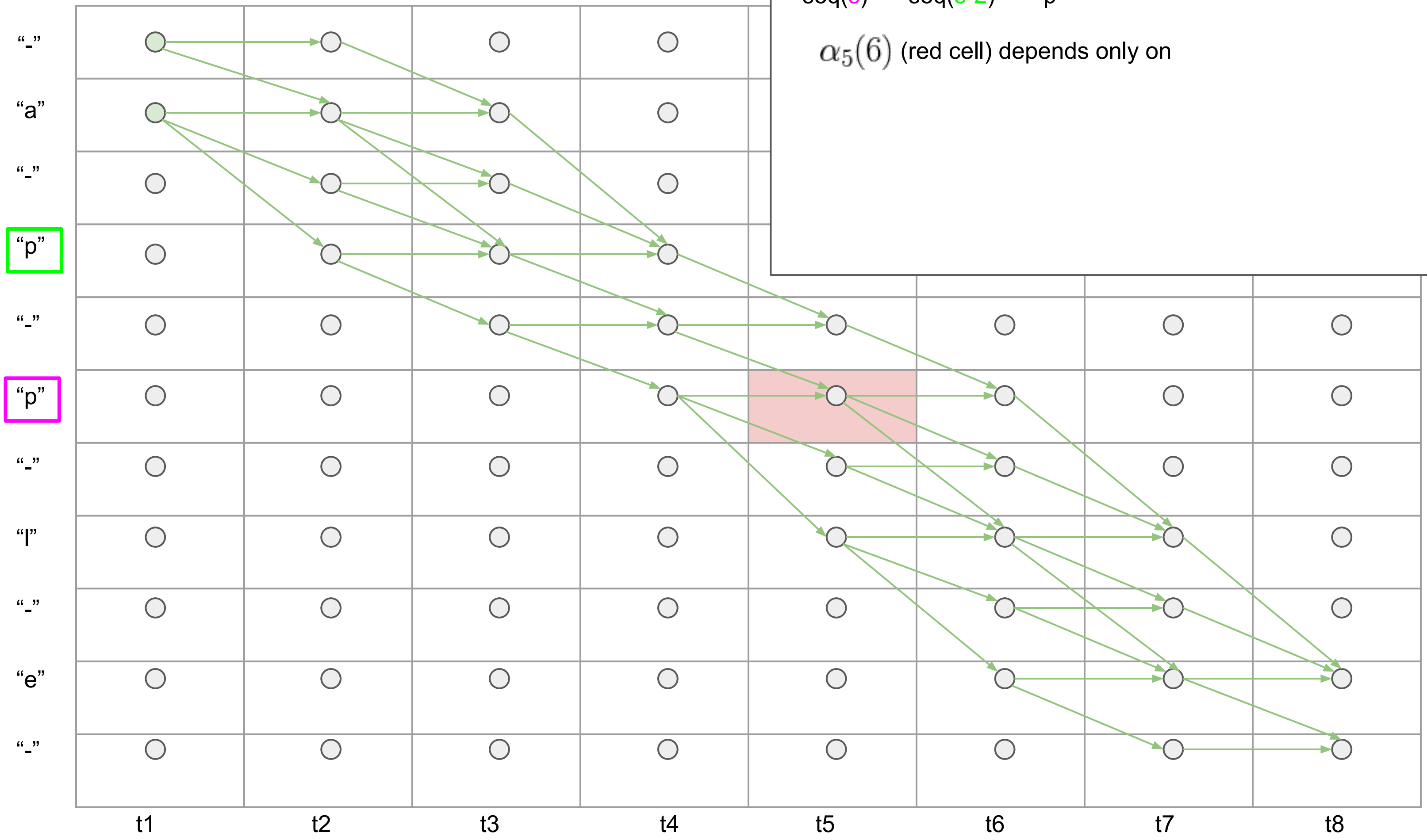
Case2. s-th symbol is equal to (s-2)-th symbol. Example: s=6,t=5

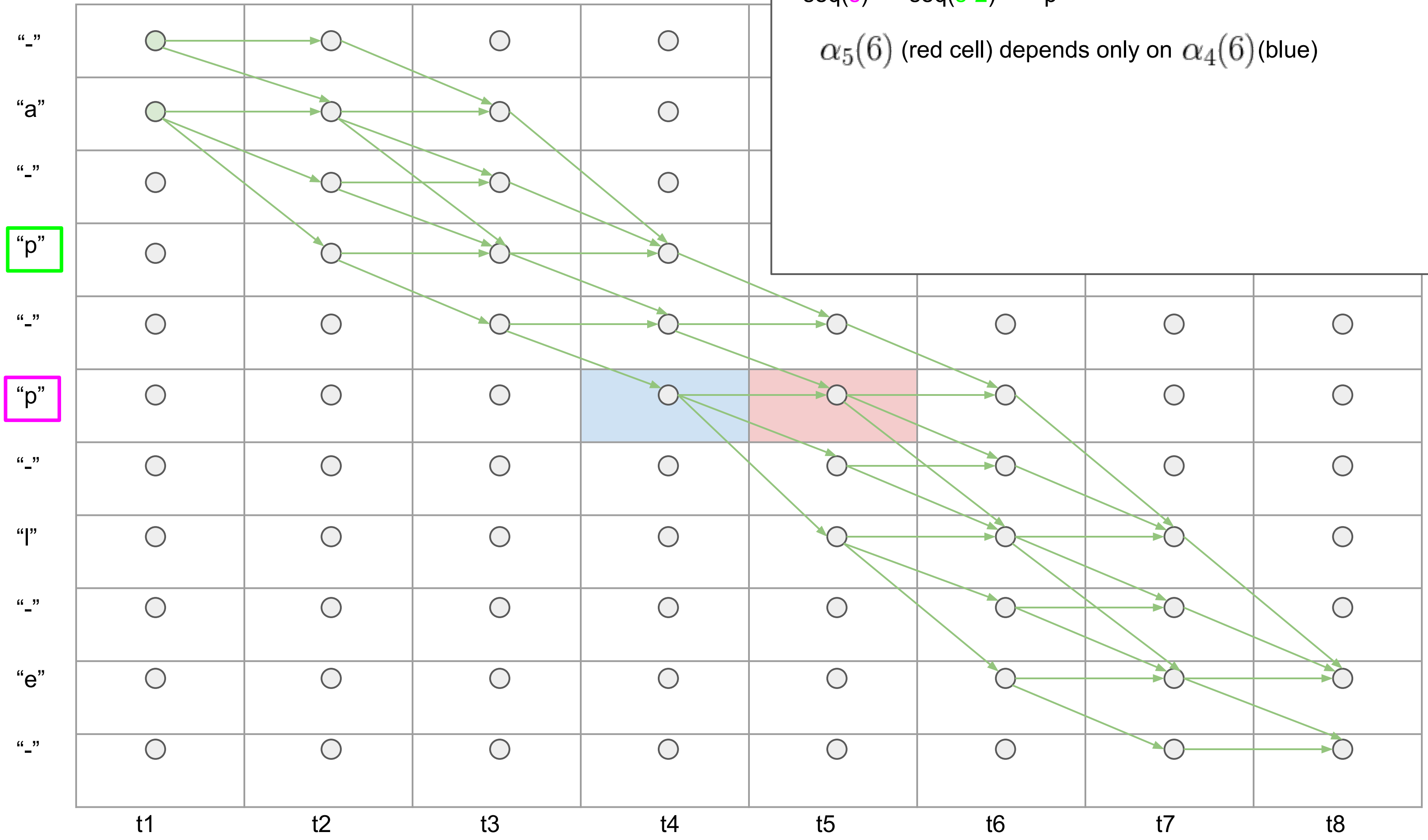
Case2. s-th symbol is equal to (s-2)-th symbol. Example: s=6,t=5.
seq(s) == seq(s-2) == “p”



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





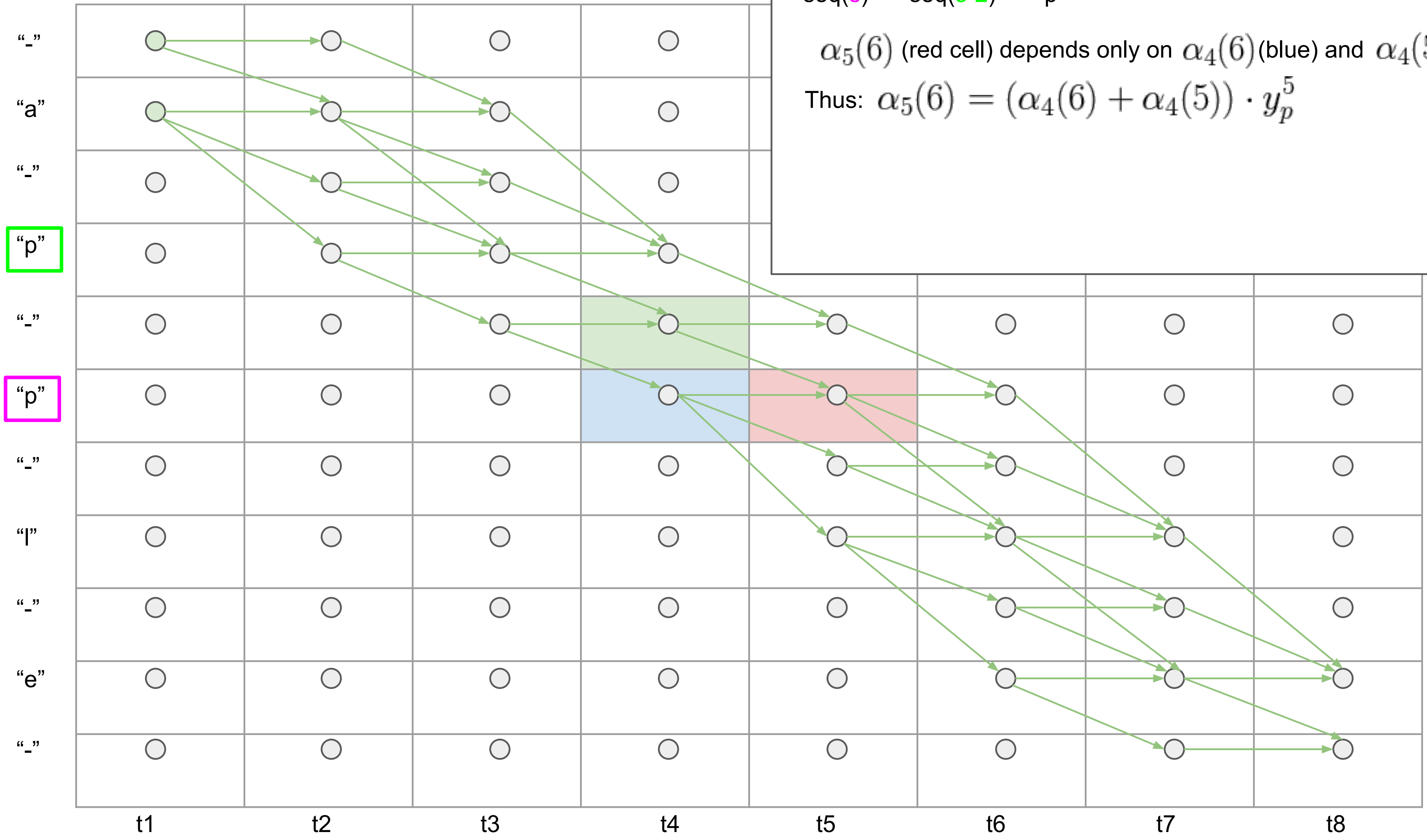
Case2. s-th symbol is equal to (s-2)-th symbol. Example: s=6,t=5.
seq(s) == seq(s-2) == “p”

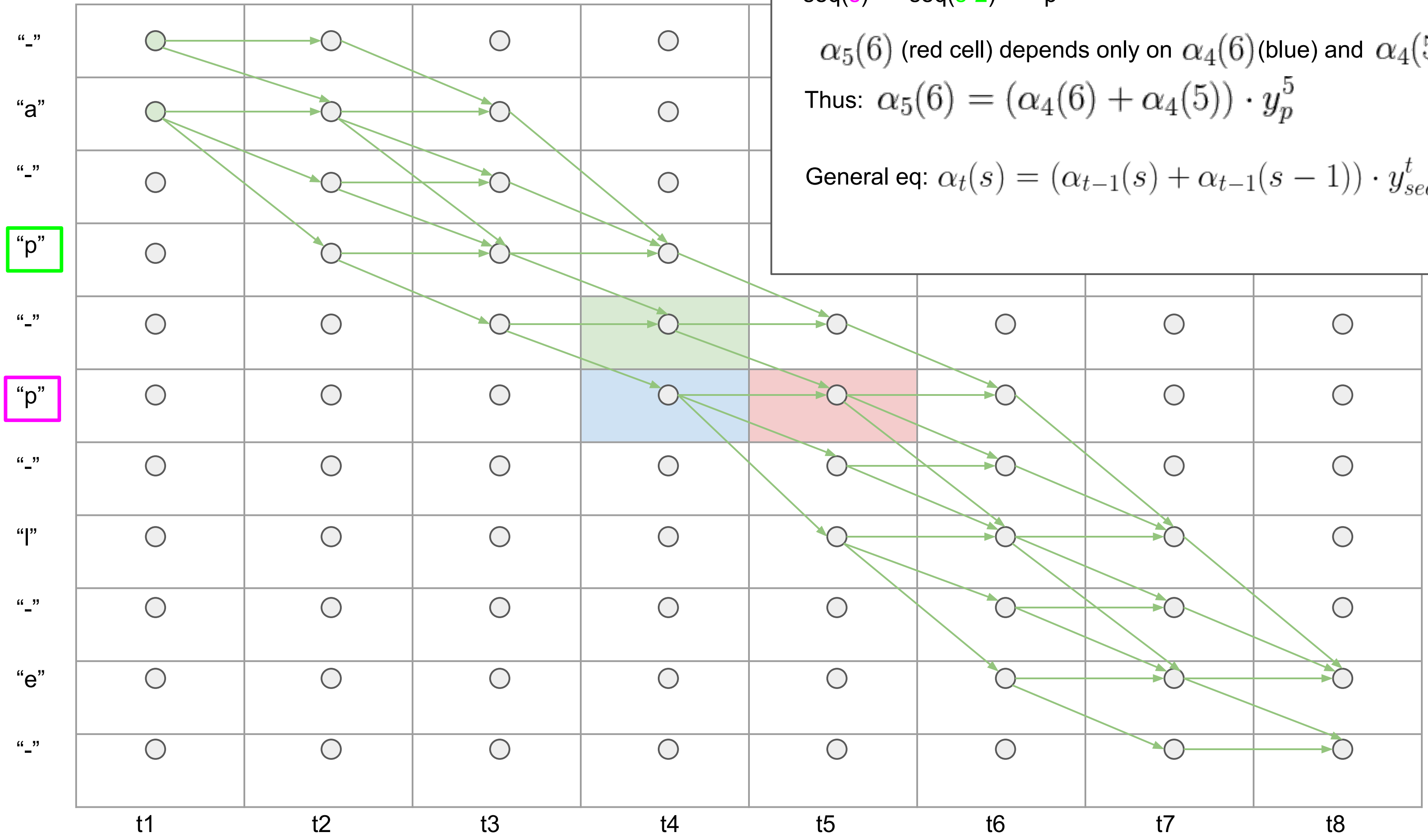
$\alpha_5(6)$ (red cell) depends only on $\alpha_4(6)$(blue)

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: $\alpha_5(6) = (\alpha_4(6) + \alpha_4(5)) \cdot y_p^5$



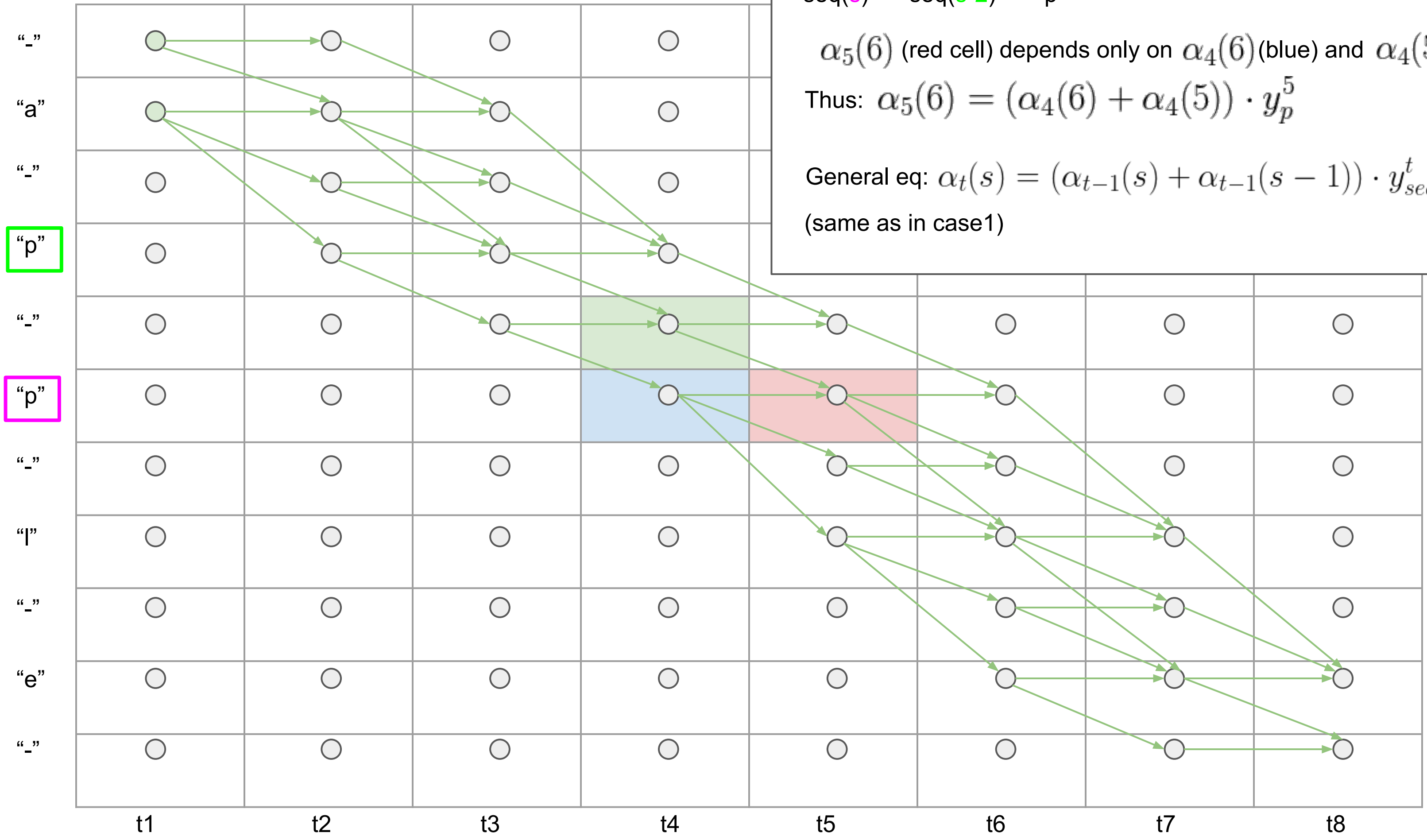


Case2. s-th symbol is equal to (s-2)-th symbol. Example: s=6,t=5.
seq(s) == seq(s-2) == “p”

$\alpha_5(6)$ (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: $\alpha_t(s) = (\alpha_{t-1}(s) + \alpha_{t-1}(s-1)) \cdot y_{seq(s)}^t$



Case2. s-th symbol is equal to (s-2)-th symbol. Example: s=6,t=5.
seq(s) == seq(s-2) == “p”

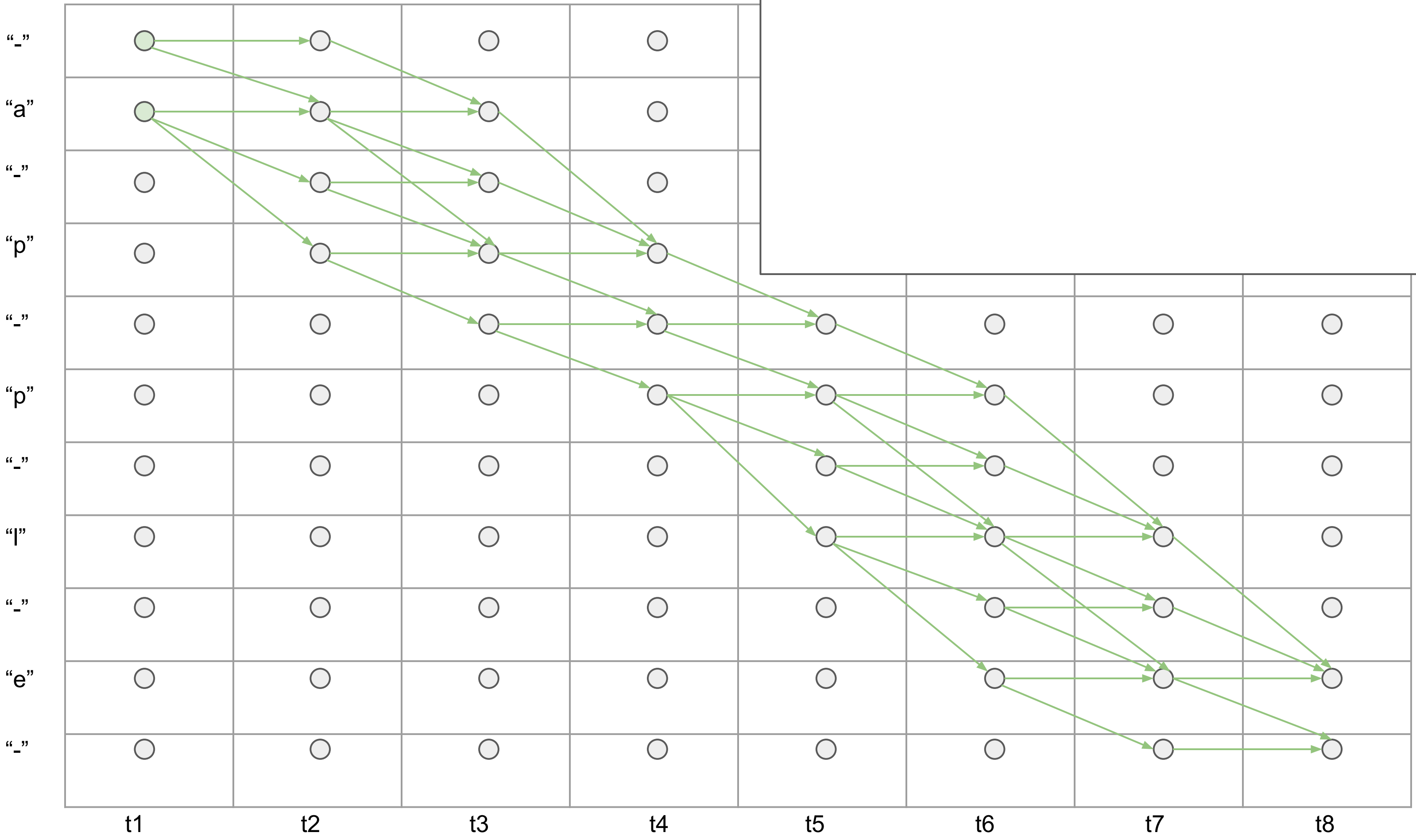
$\alpha_5(6)$ (red cell) depends only on $\alpha_4(6)$(blue) and $\alpha_4(5)$(green)

Thus: $\alpha_5(6) = (\alpha_4(6) + \alpha_4(5)) \cdot y_p^5$

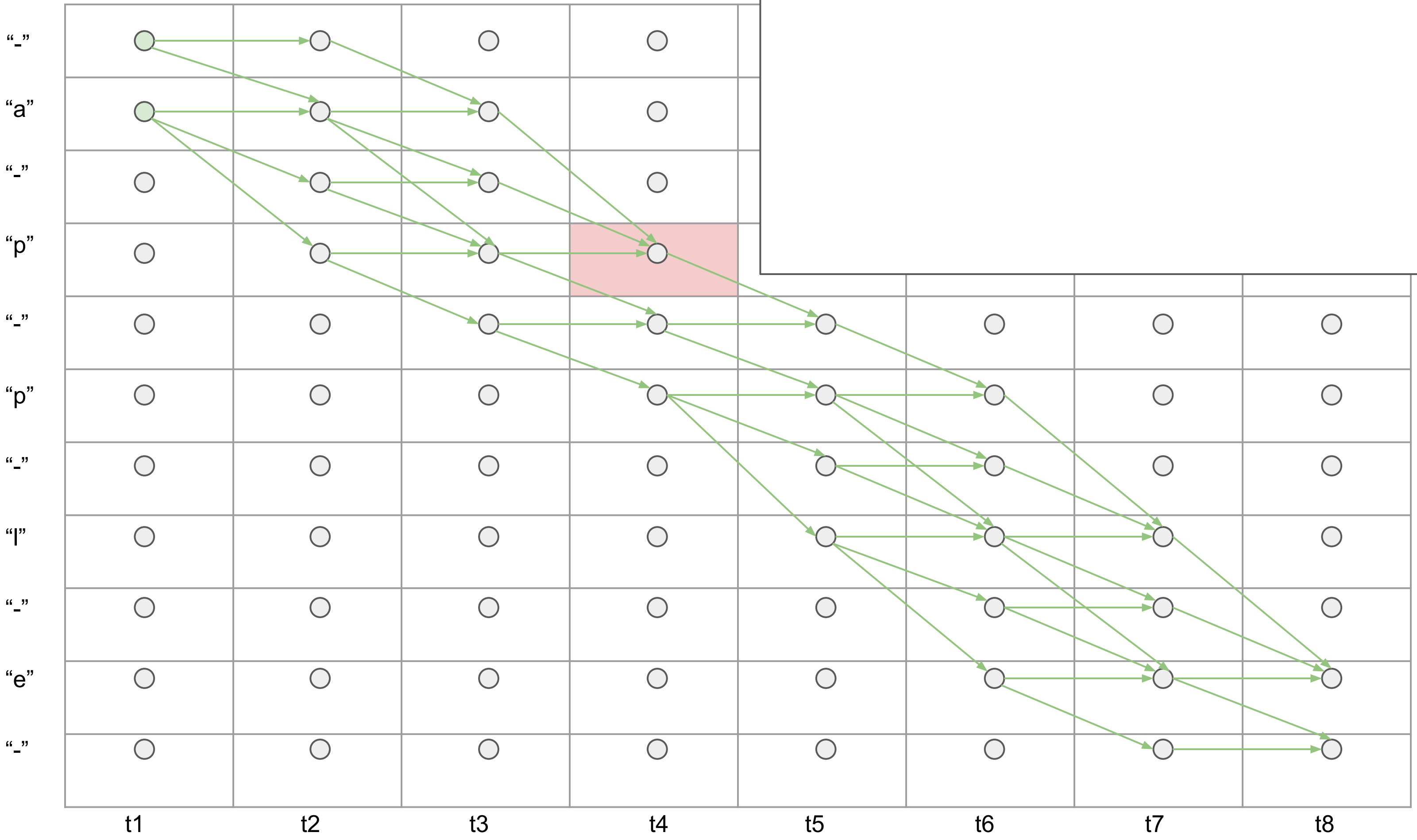
General eq: $\alpha_t(s) = (\alpha_{t-1}(s) + \alpha_{t-1}(s-1)) \cdot y_{seq(s)}^t$

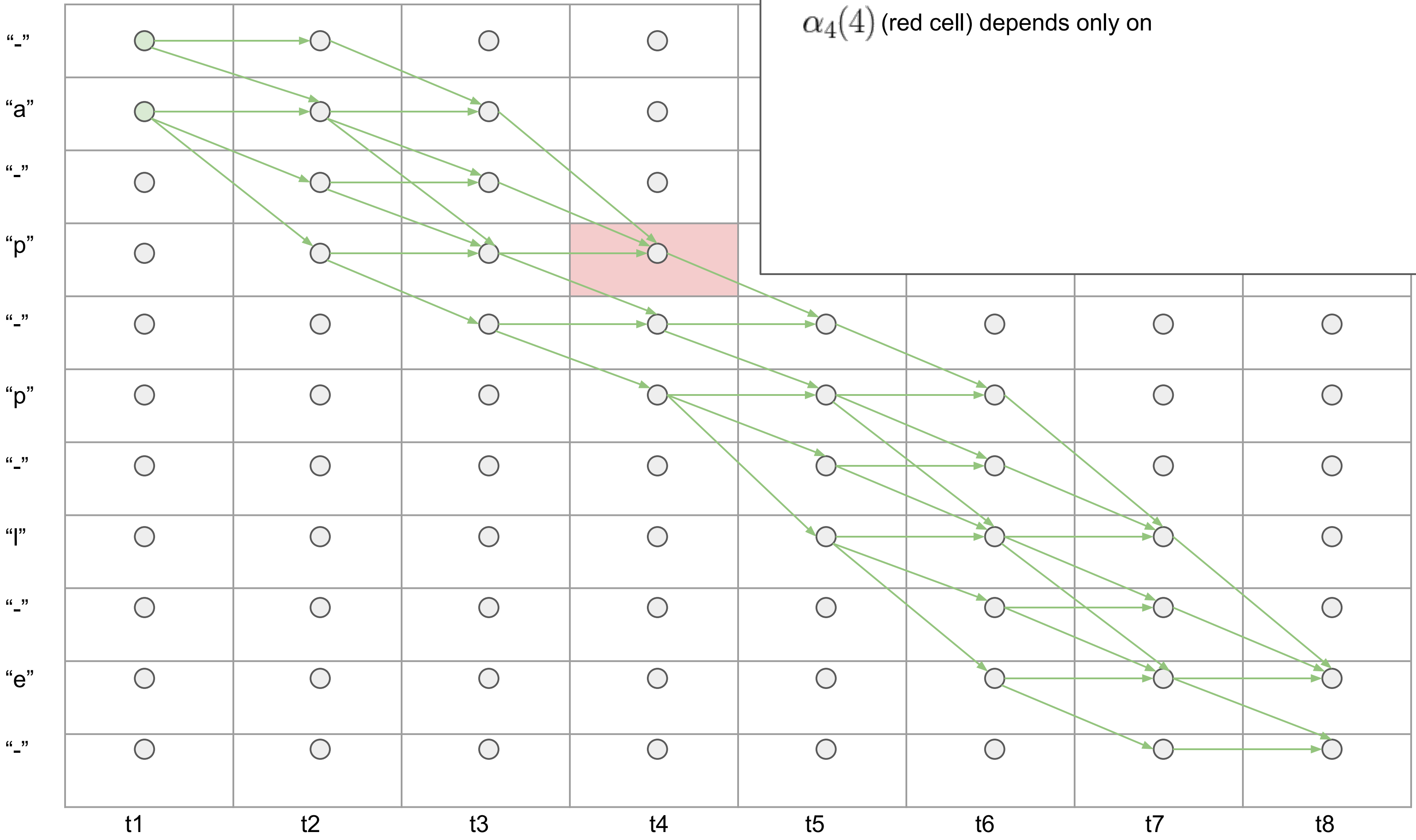
(same as in case1)

Case3. Otherwise.



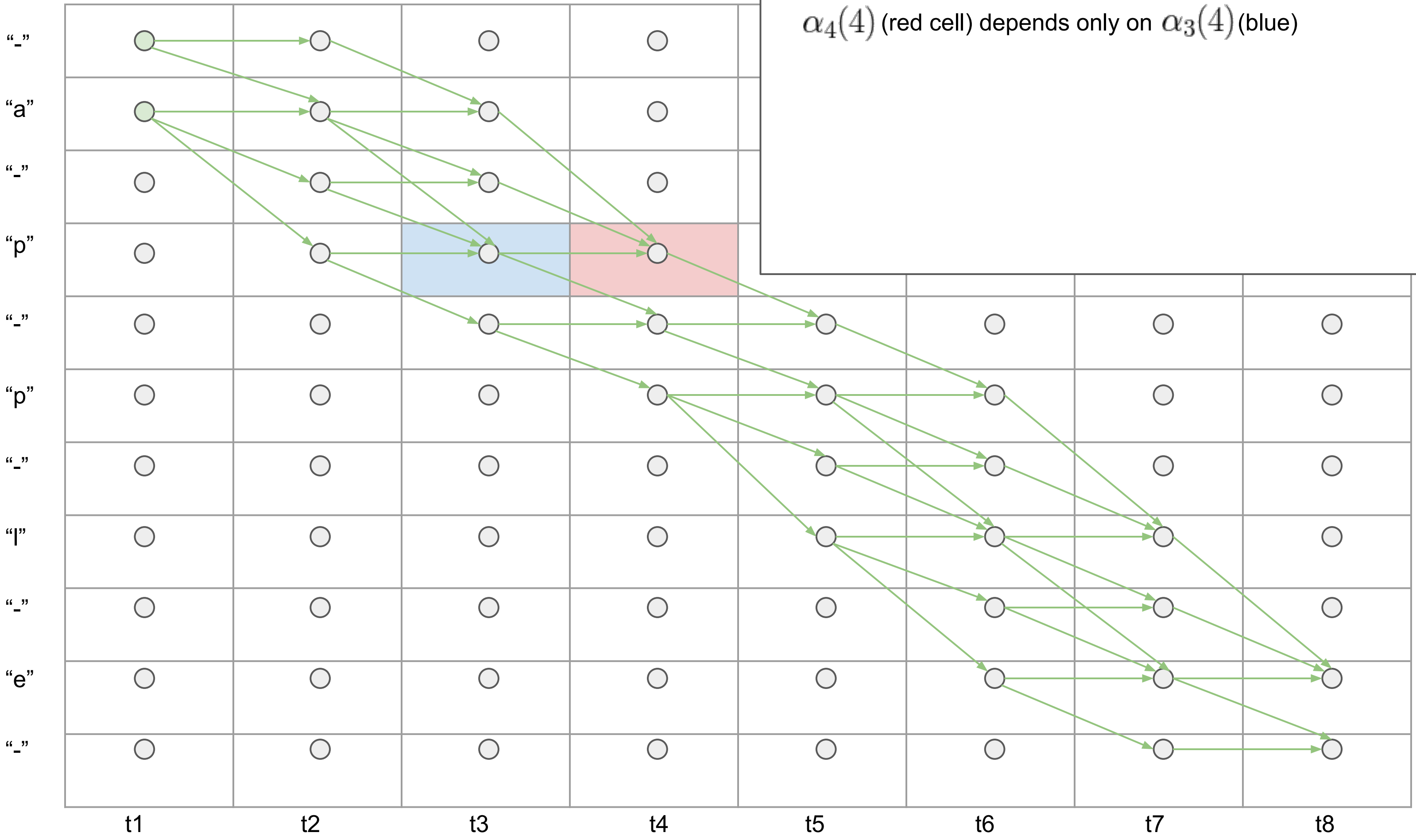
Case3. Otherwise. Example: s=4,t=4.





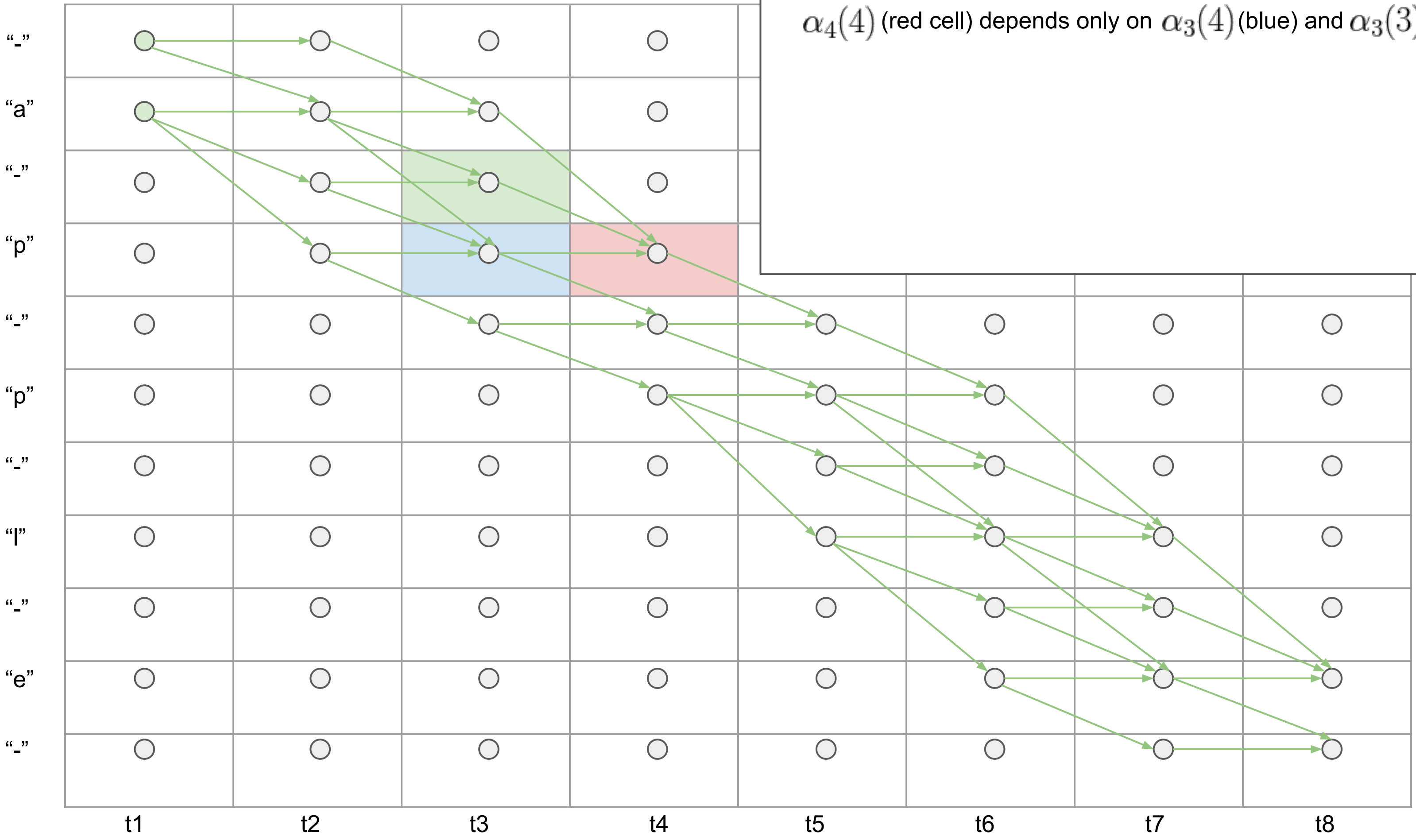
Case3. Otherwise. Example: s=4,t=4.

$\alpha_4(4)$ (red cell) depends only on



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

$\alpha_4(4)$ (red cell) depends only on $\alpha_3(4)$ (blue)

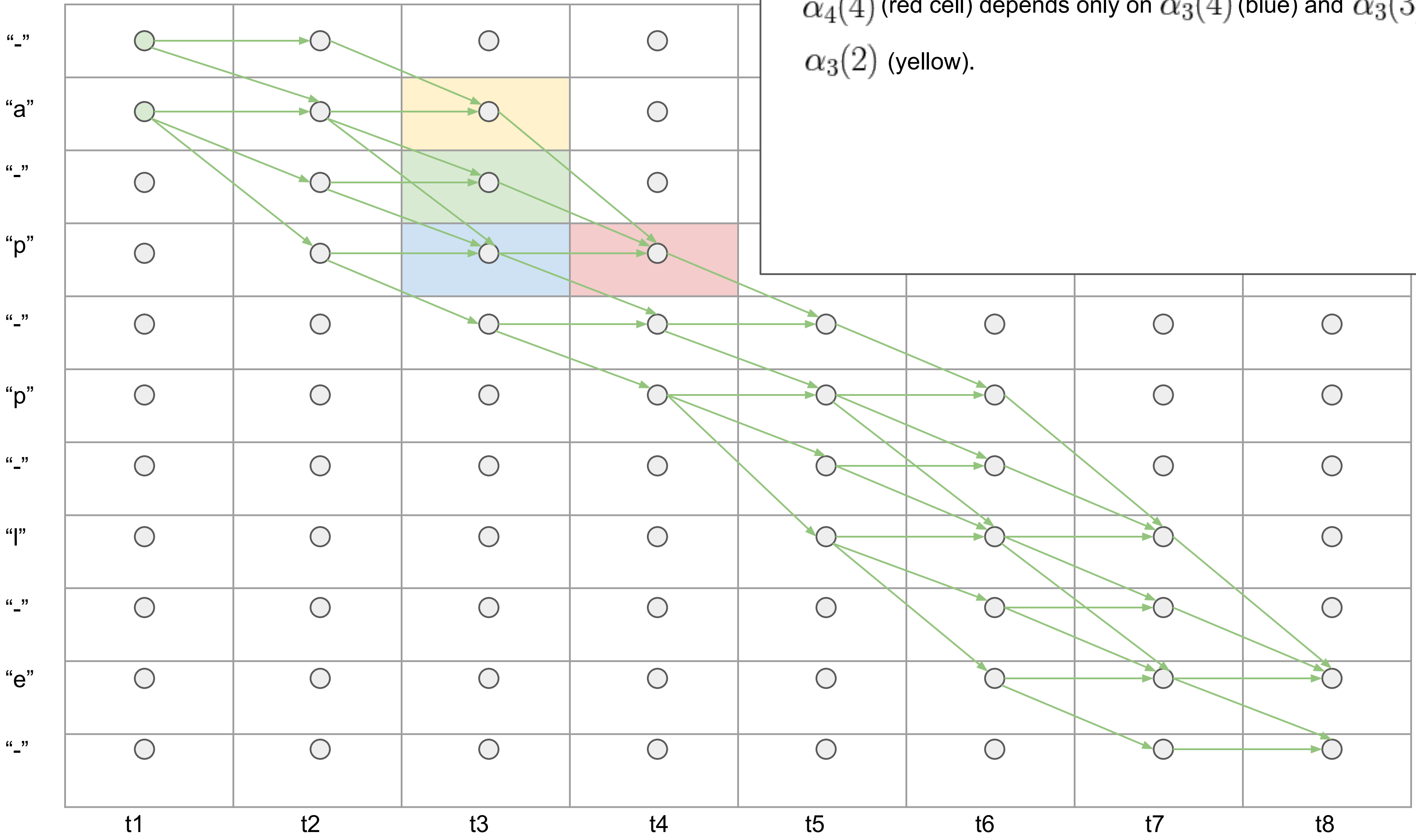


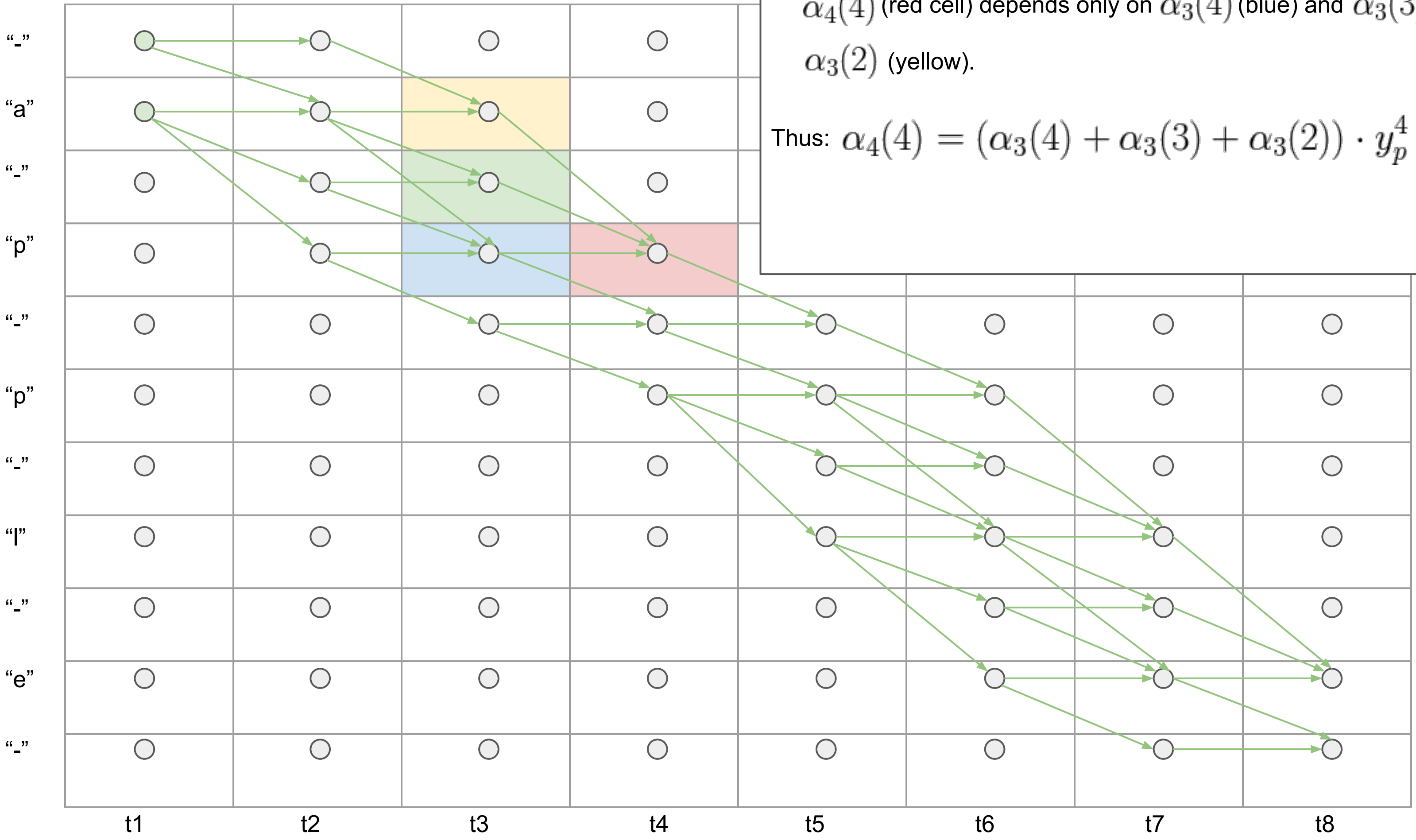
Case3. Otherwise. Example: s=4,t=4.

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

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).

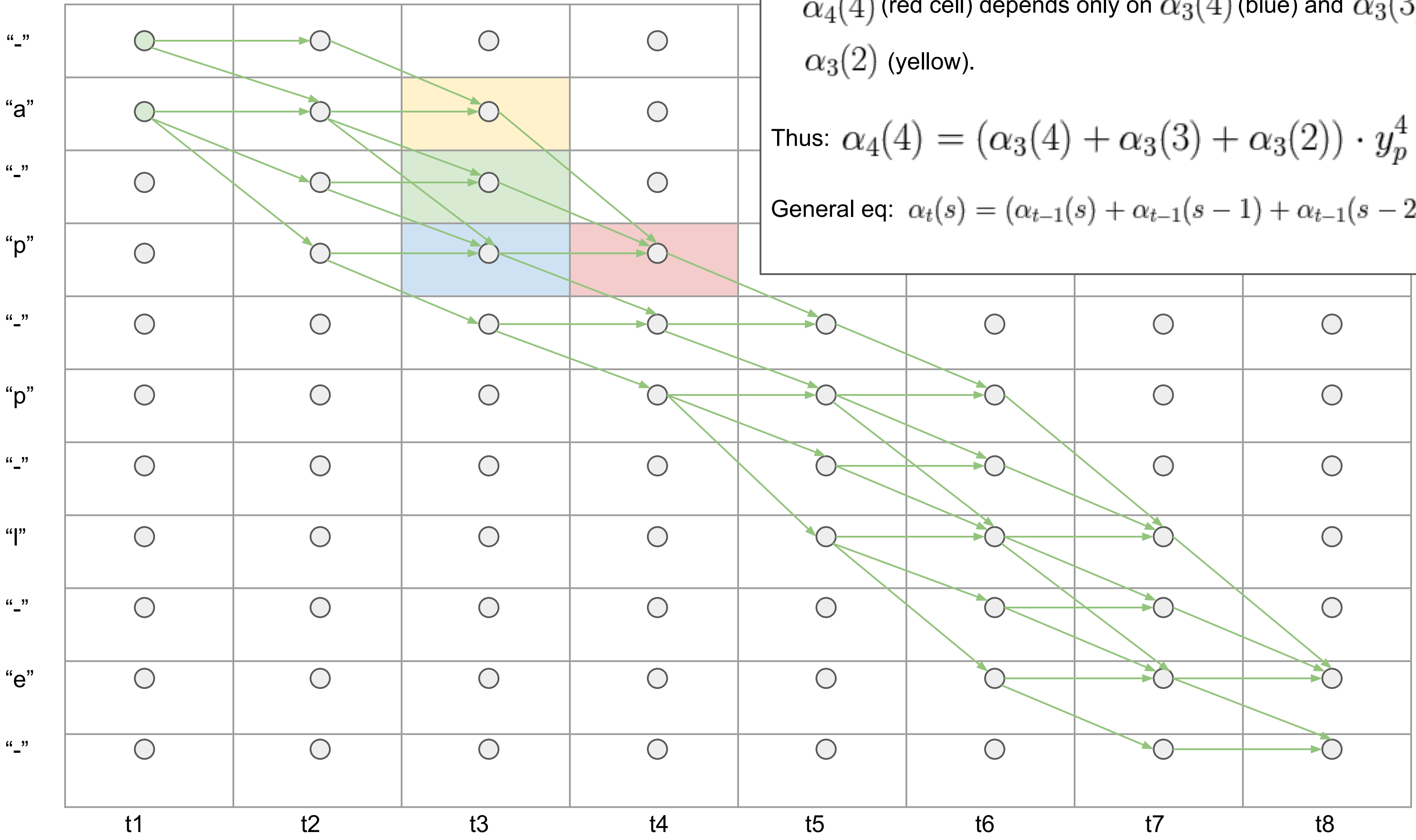




Case3. Otherwise. Example: s=4,t=4.

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

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

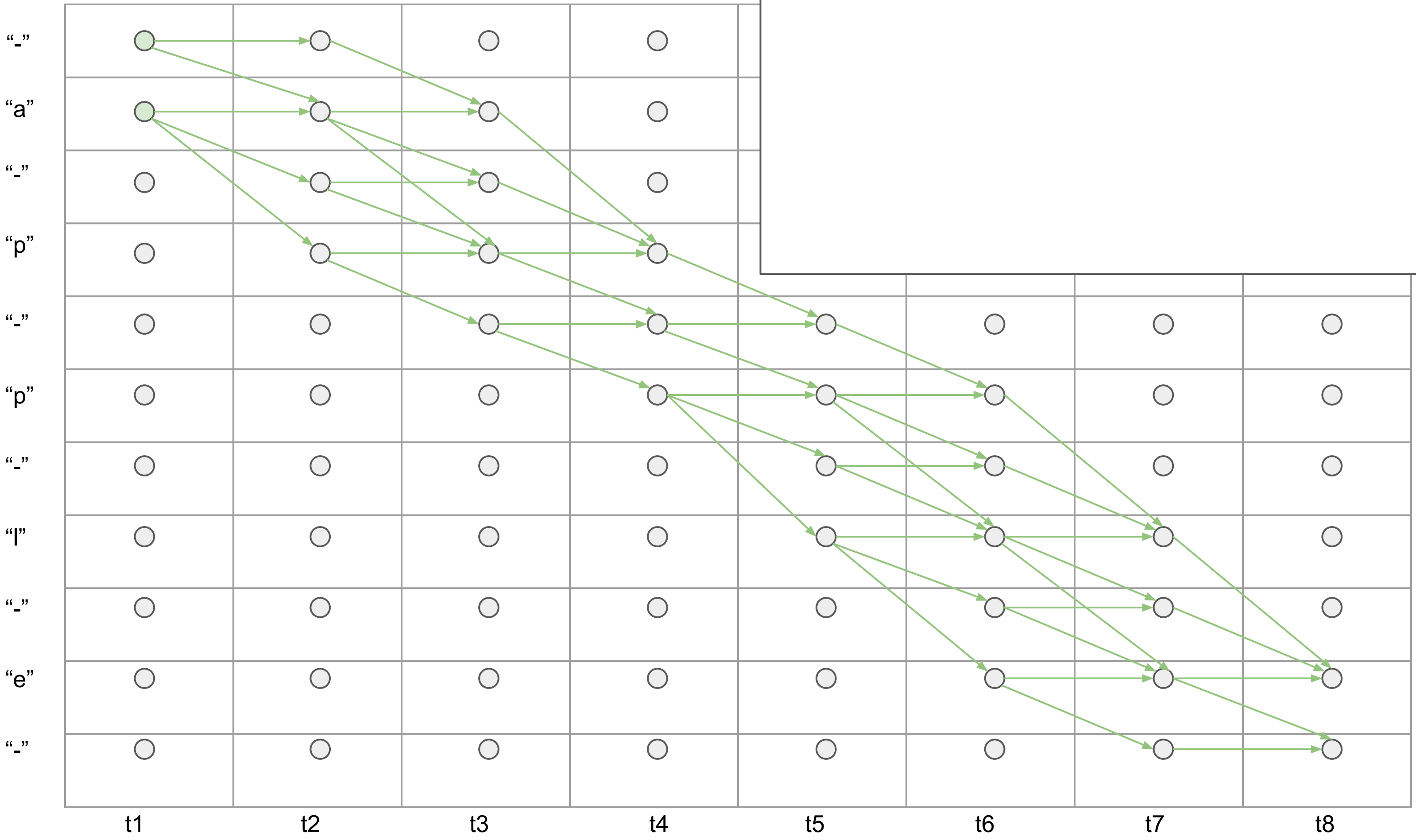


Case3. Otherwise. Example: s=4,t=4.

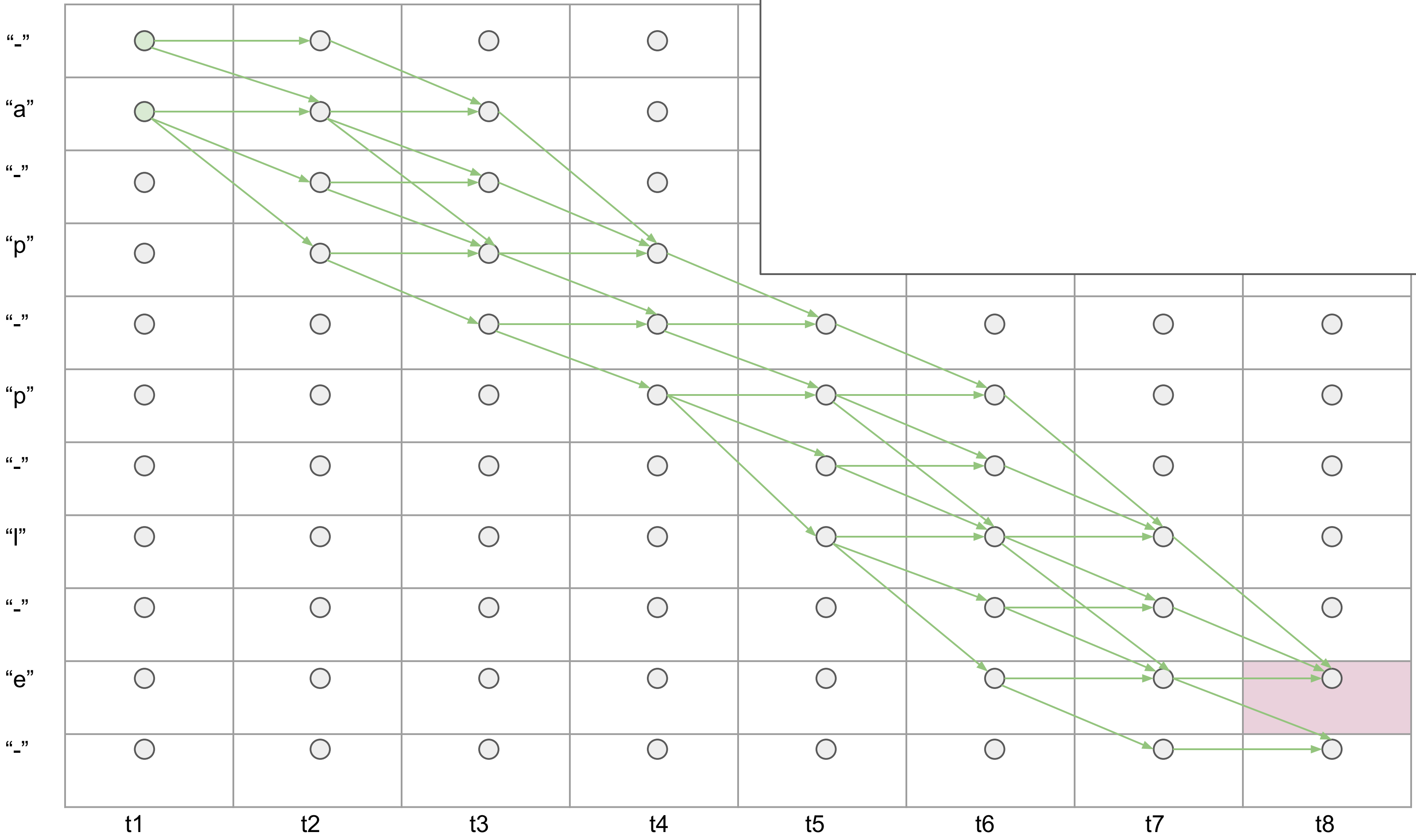
$\alpha_4(4)$ (red cell) depends only on $\alpha_3(4)$ (blue) and $\alpha_3(3)$ (green) and $\alpha_3(2)$ (yellow).

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

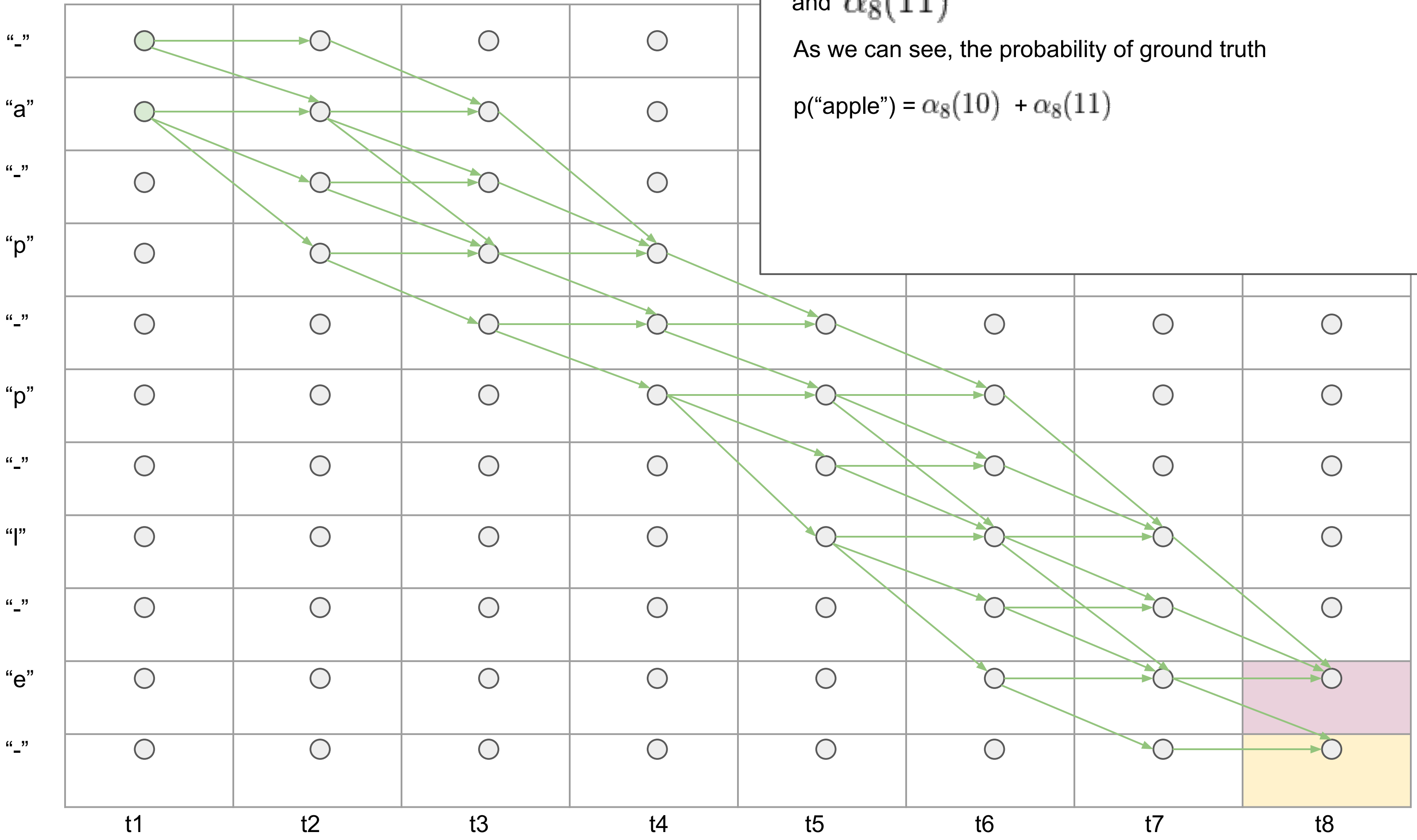
General eq: $\alpha_t(s) = (\alpha_{t-1}(s) + \alpha_{t-1}(s-1) + \alpha_{t-1}(s-2)) \cdot y_{seq(s)}^t$



With this dynamic programming algorithm we can calculate



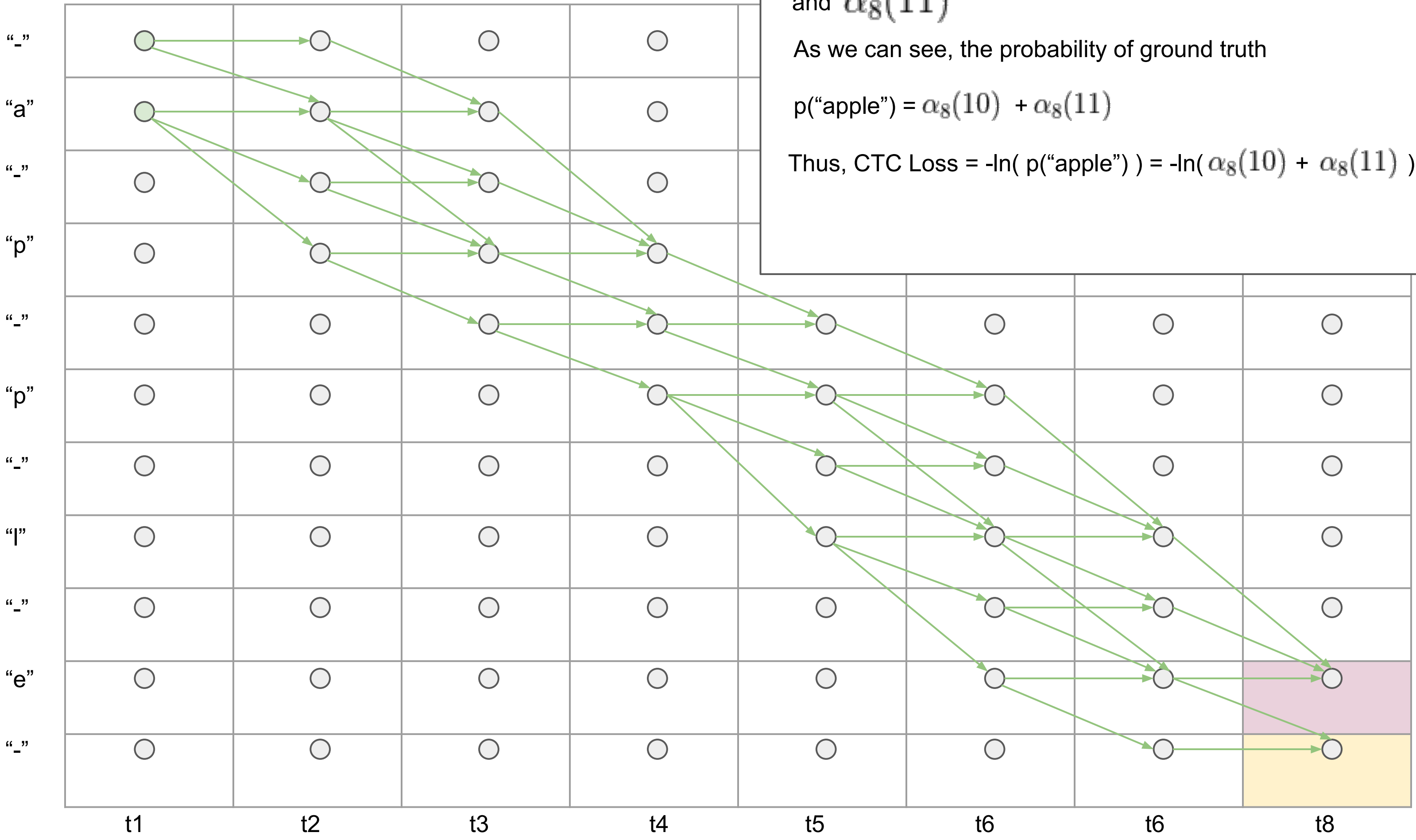
With this dynamic programming algorithm we can calculate $\alpha_8(10)$



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

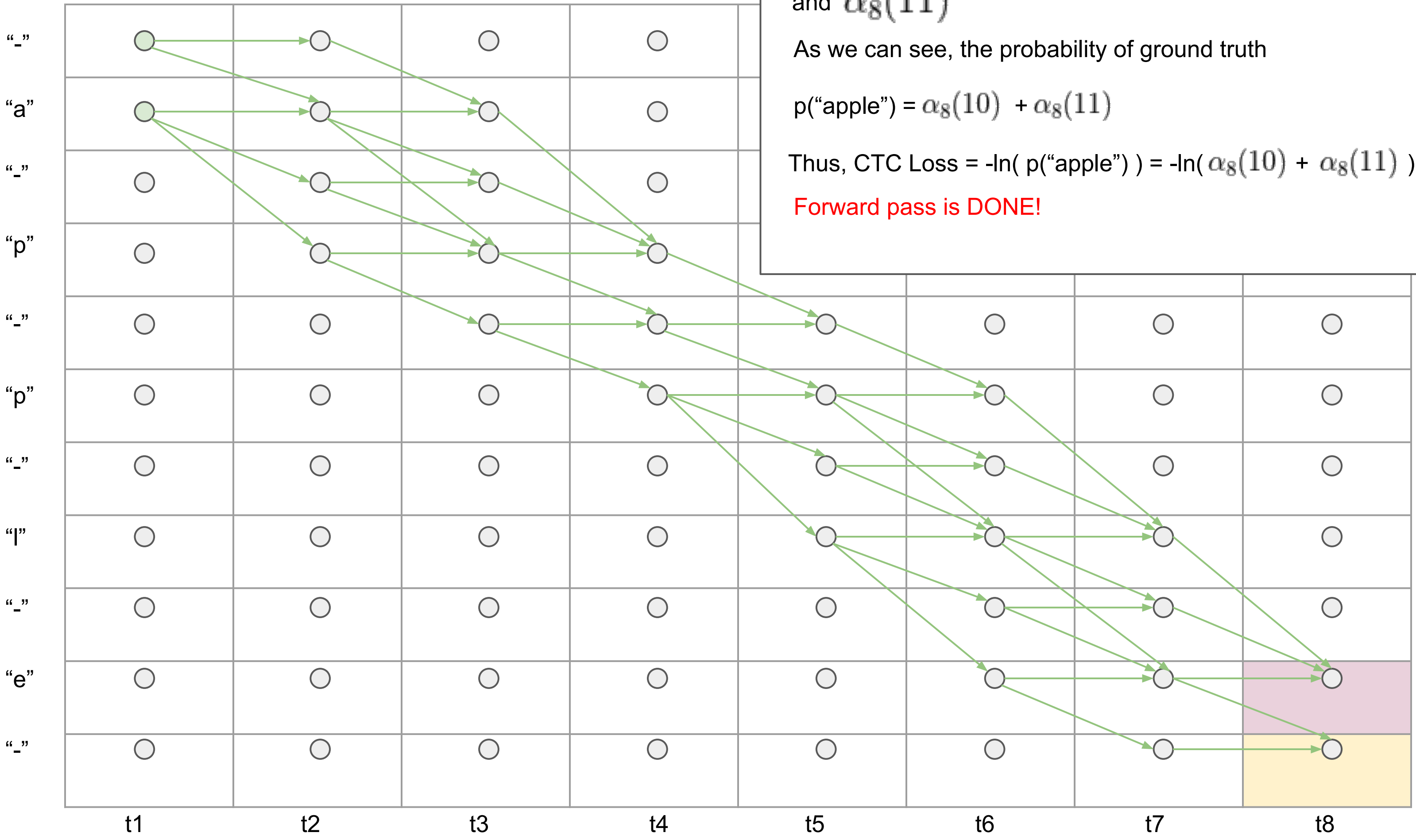


With this dynamic programming algorithm we can calculate $\alpha_8(10)$ and $\alpha_8(11)$

As we can see, the probability of ground truth

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

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



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!

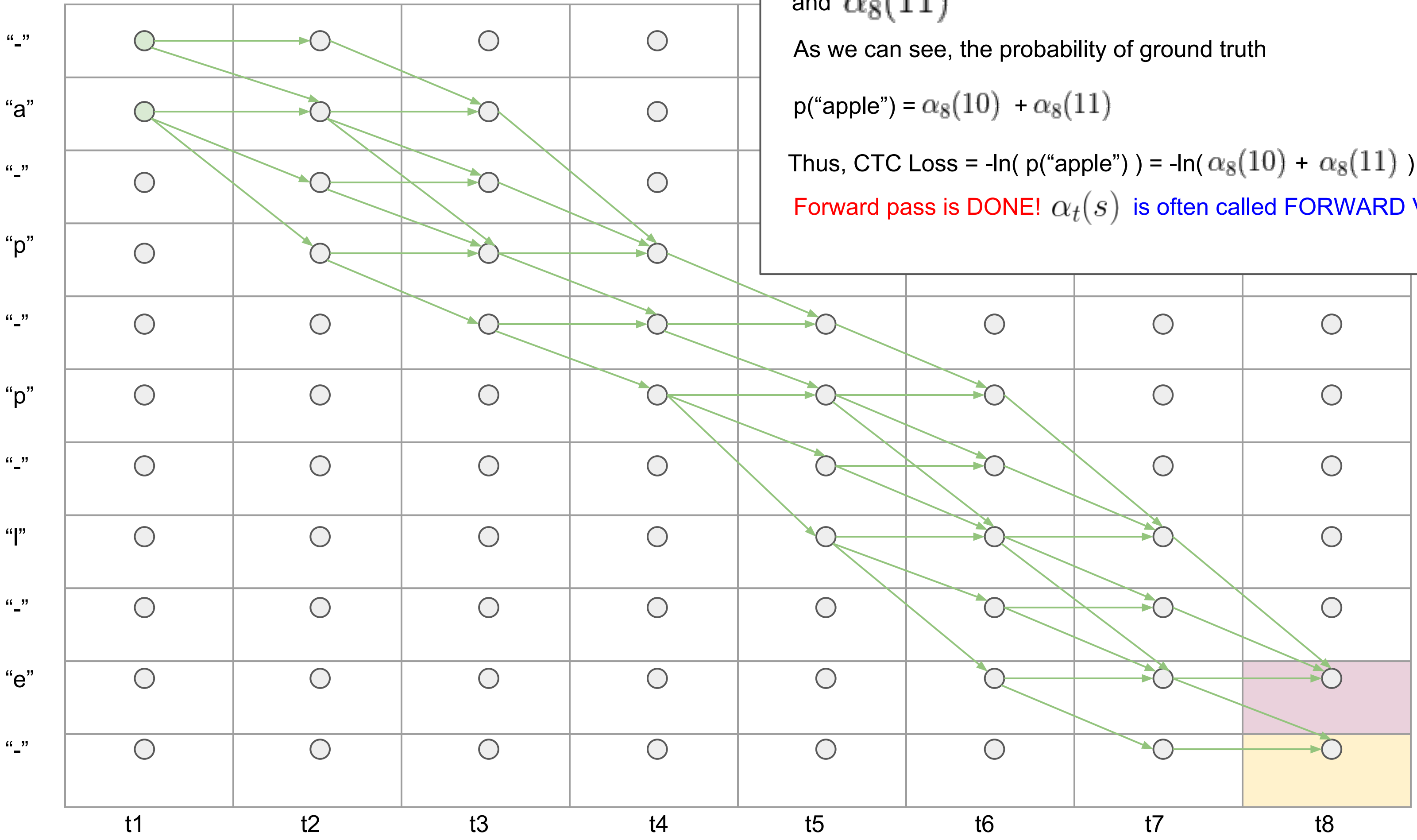
With this dynamic programming algorithm we can calculate $\alpha_8(10)$ and $\alpha_8(11)$

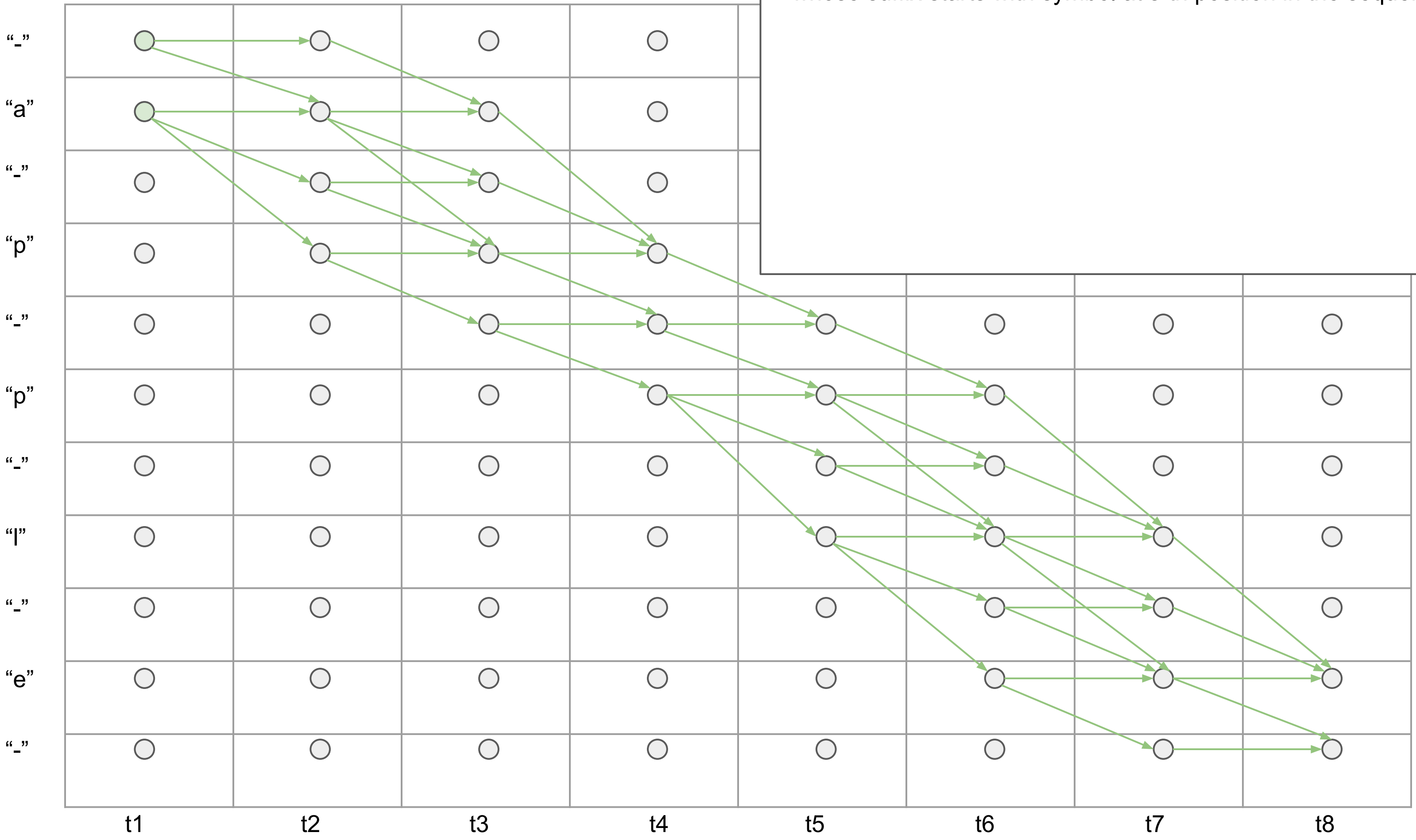
As we can see, the probability of ground truth

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.

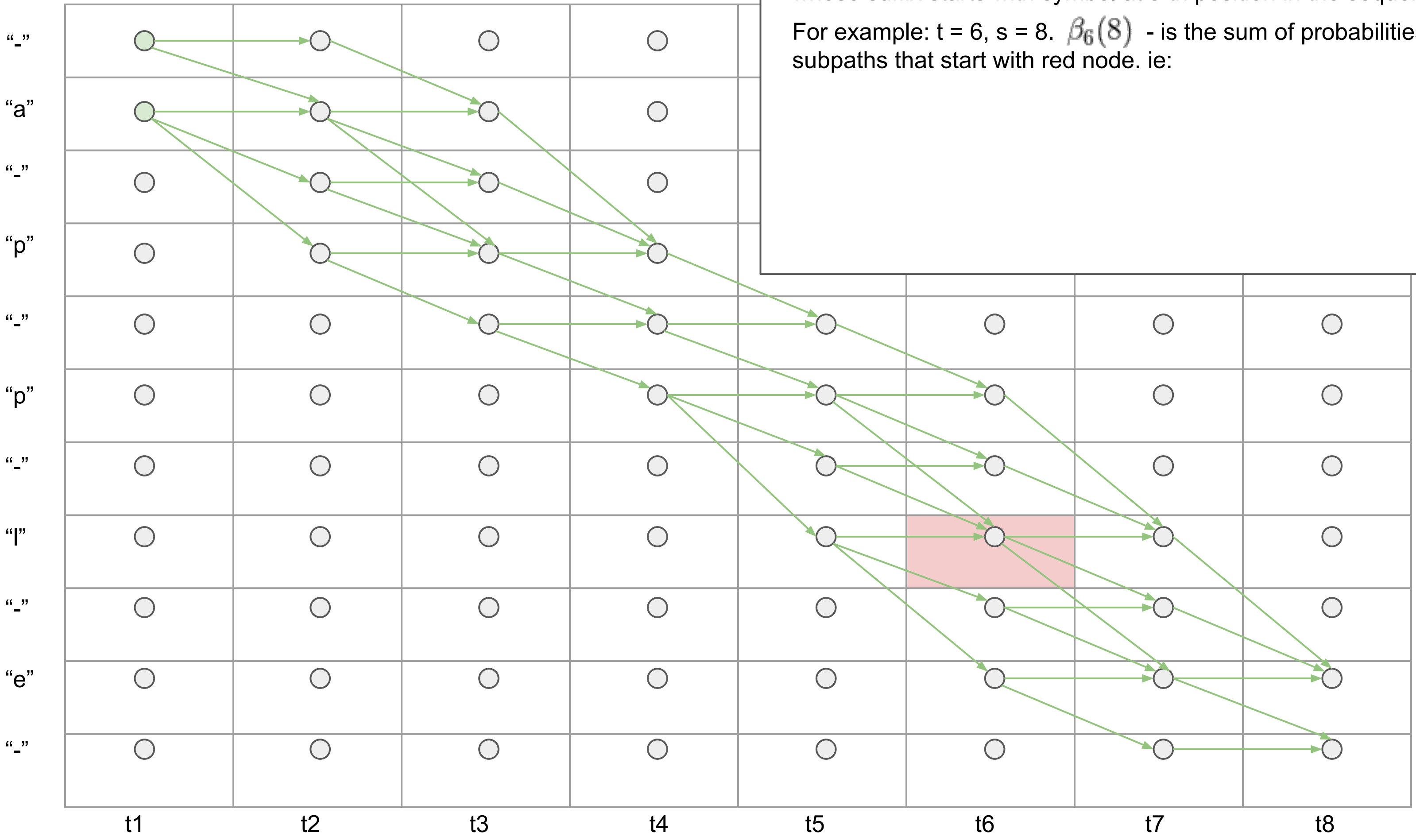




Let me define variable $\beta_t(s)$ - the total probability of all subpaths, whose suffix starts with symbol at s-th position in the sequence at time t.

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.

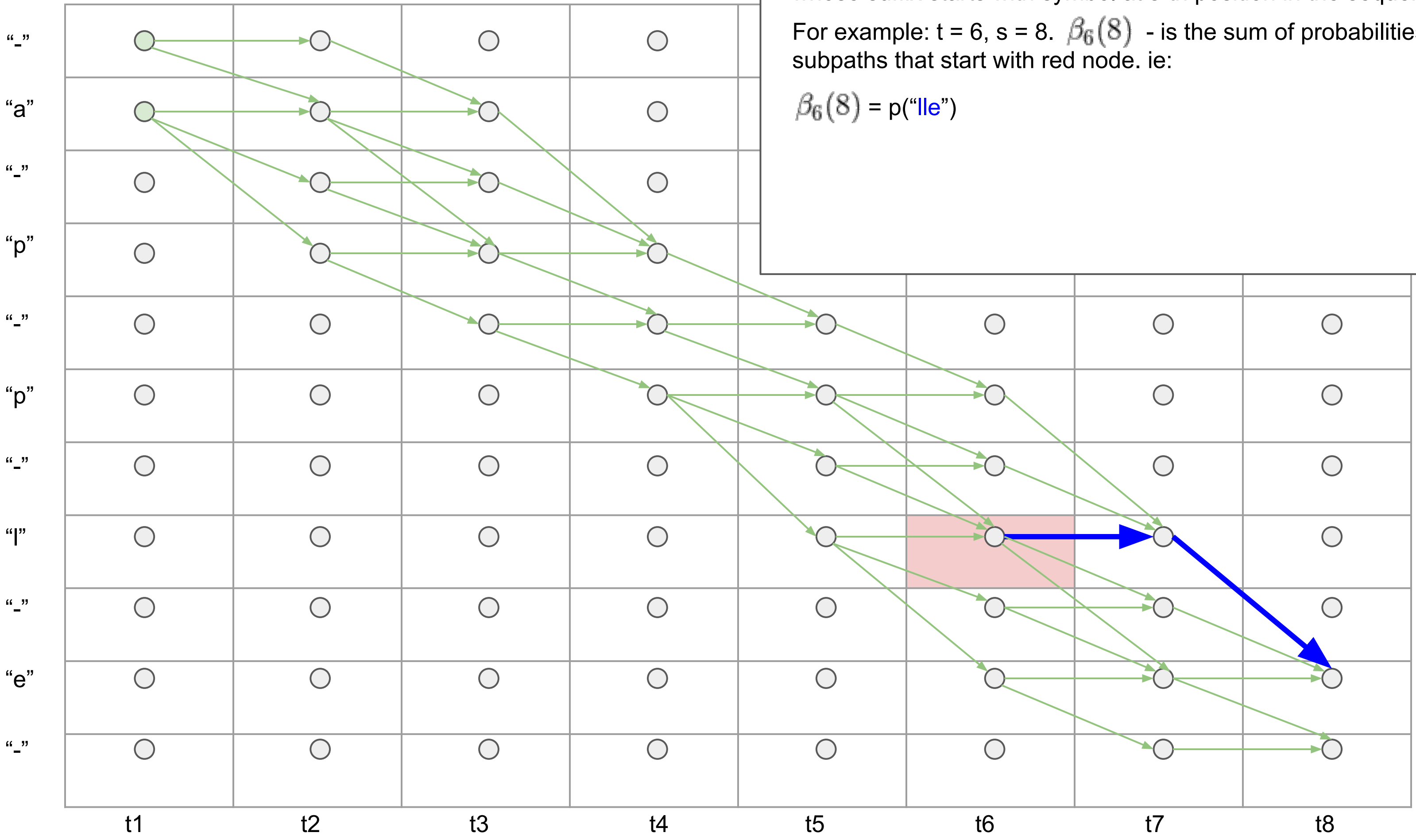
For example: t = 6, s = 8. $\beta_6(8)$ - is the sum of probabilities of all subpaths that start with red node. ie:



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:

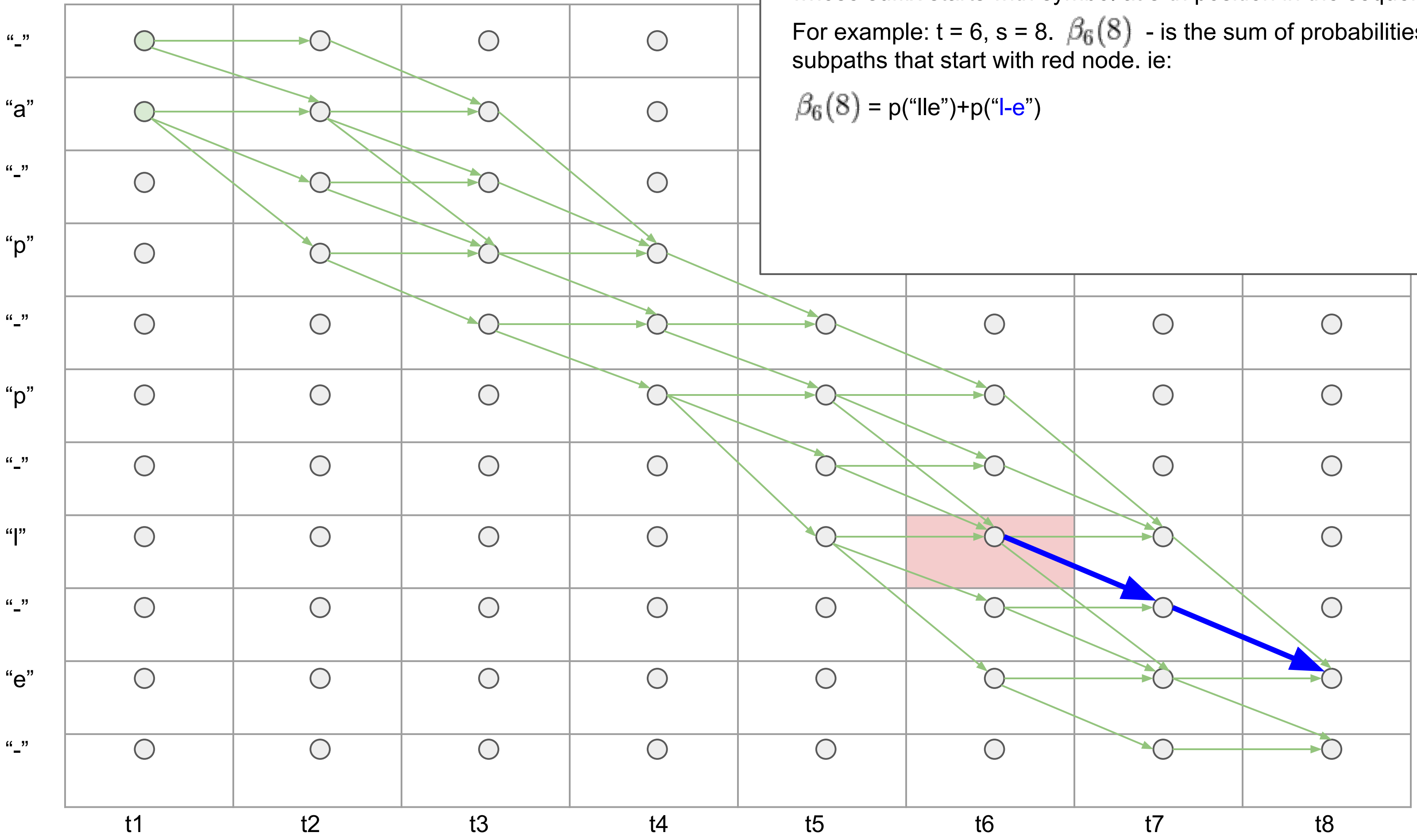
$\beta_6(8)$ = p(“lle”)



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:

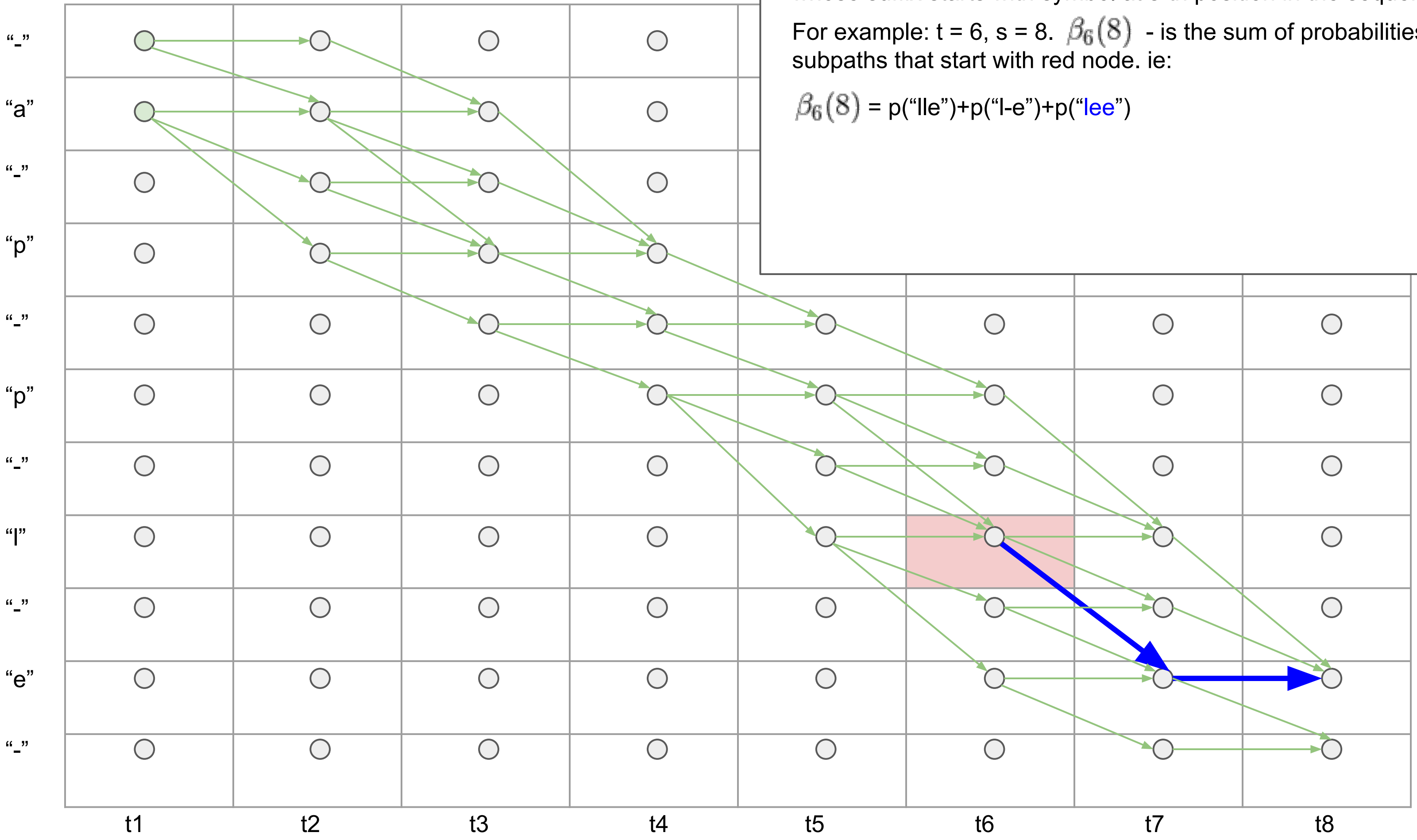
$\beta_6(8)$ = p("lle")+p("l-e")



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:

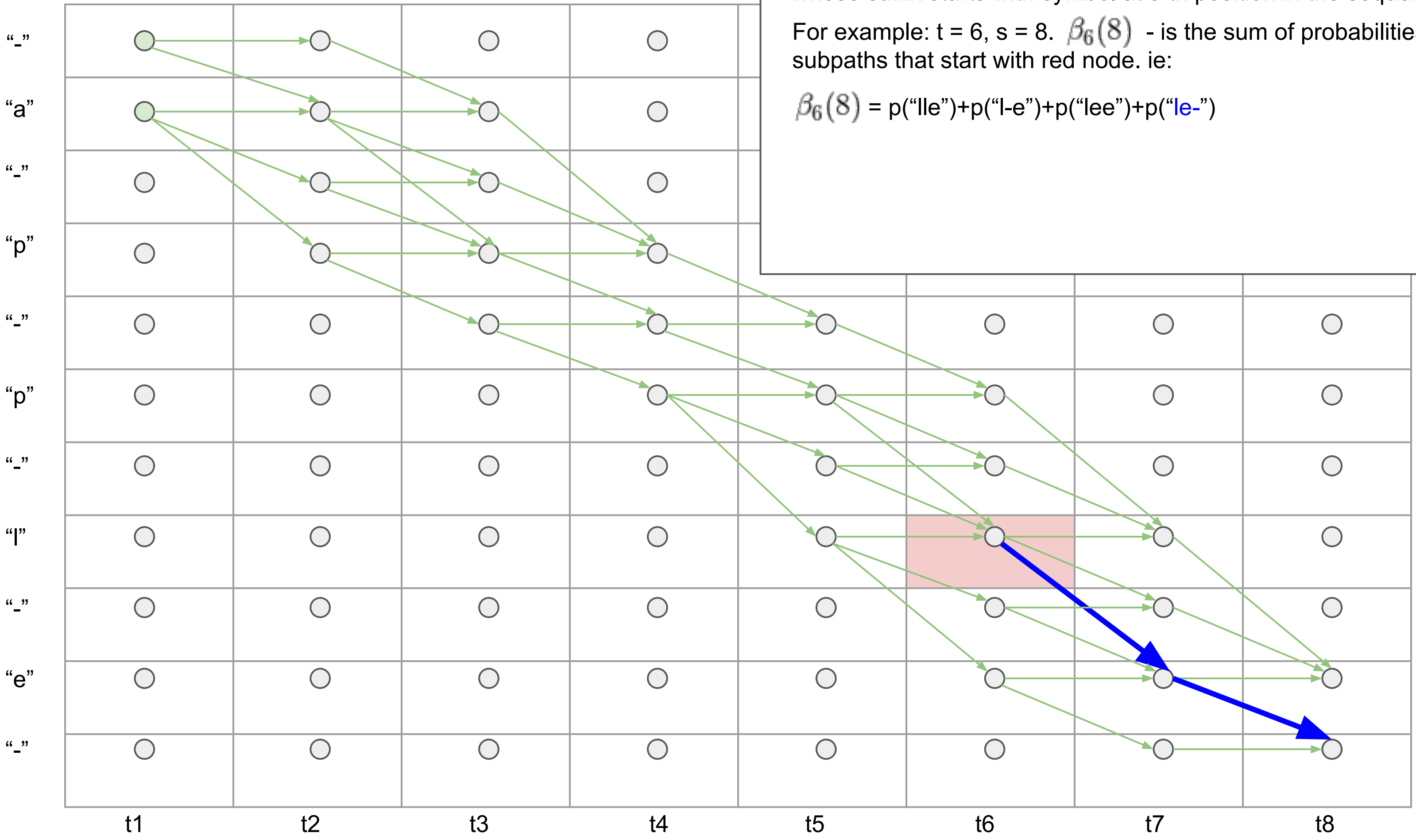
$\beta_6(8)$ = p(“lle”)+p(“l-e”)+p(“lee”)



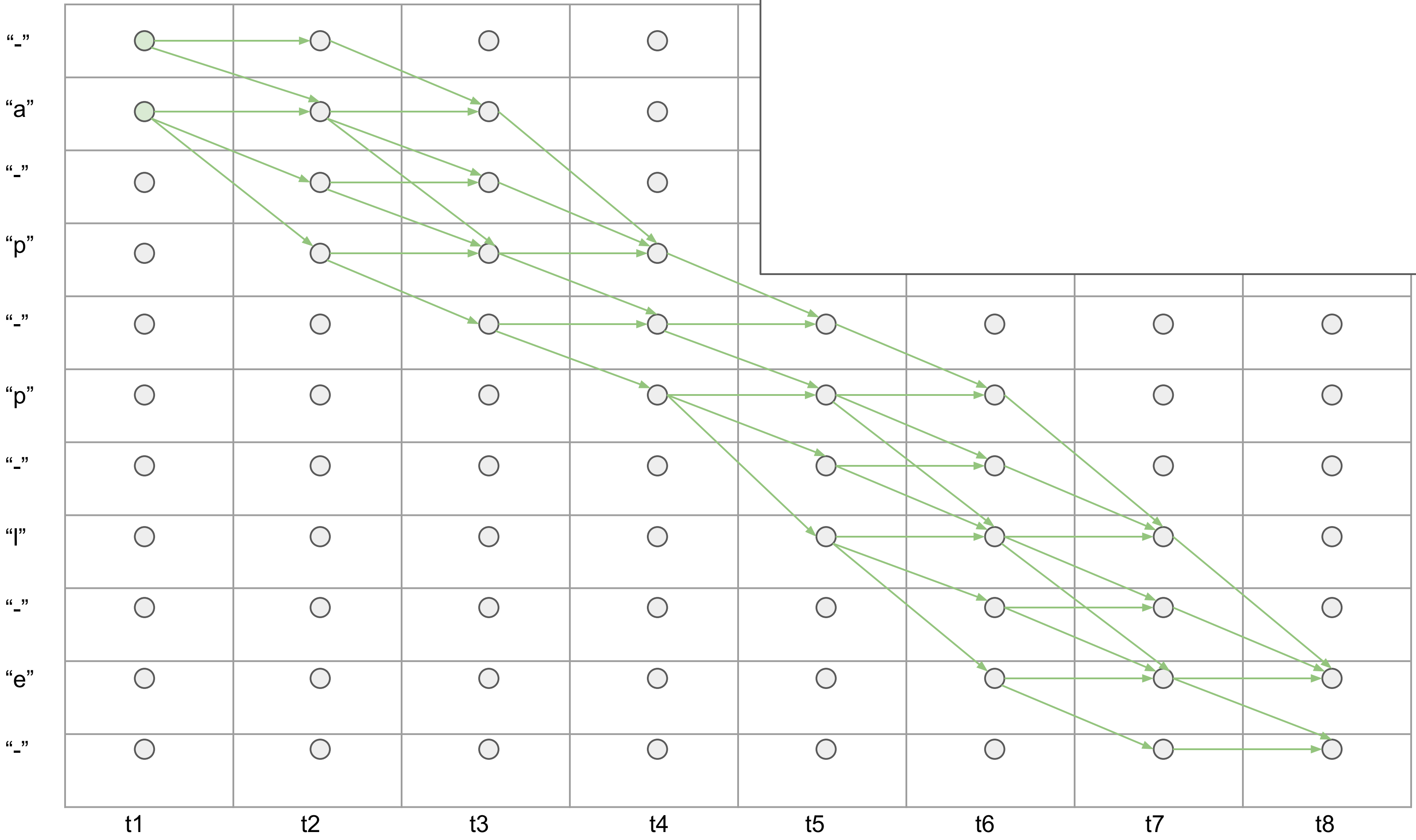
Let me define variable $\beta_t(s)$ - the total probability of all subpaths, whose suffix starts with symbol at s-th position in the sequence at time t.

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-”)

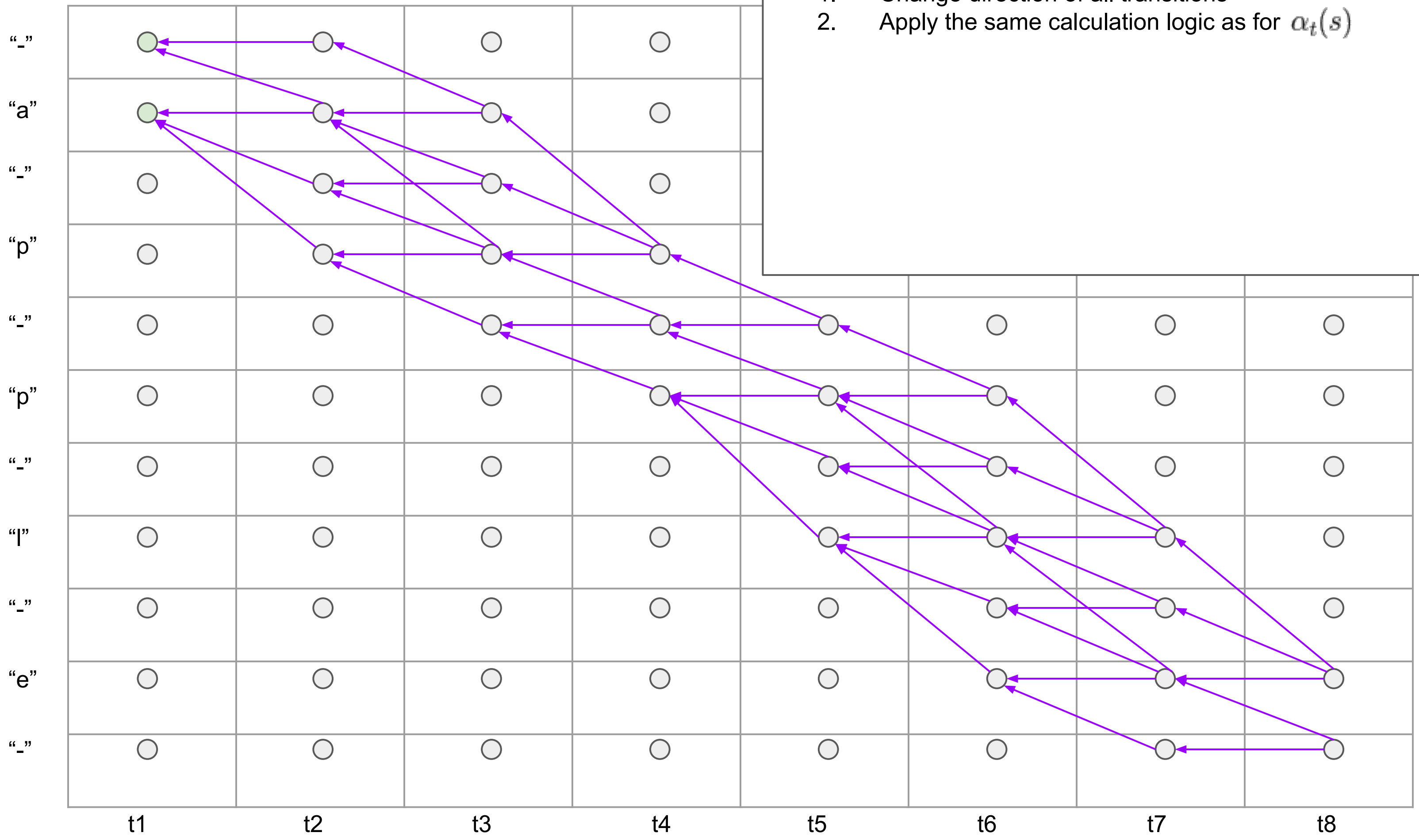


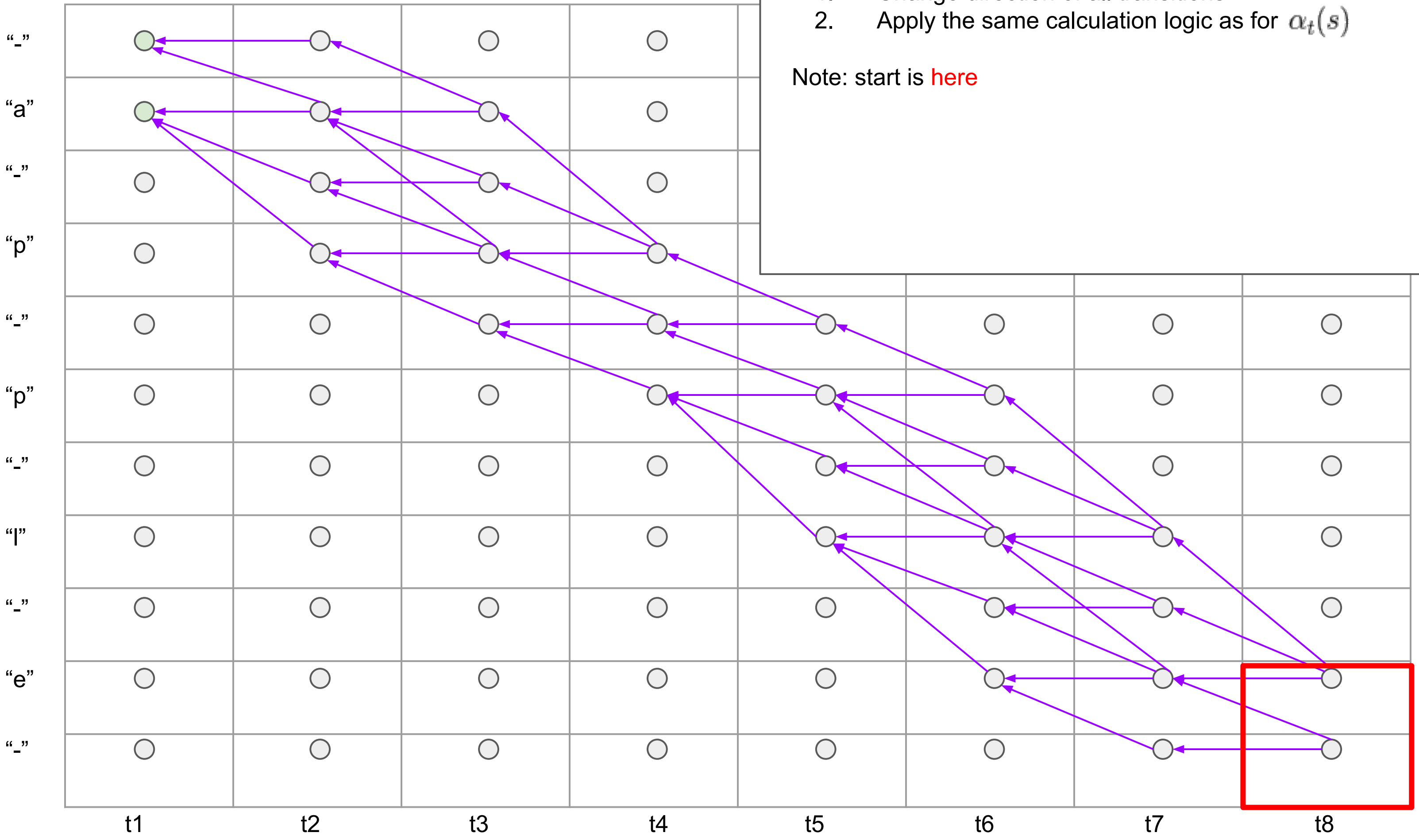
To calculate variable $\beta_t(s)$ we should:



To calculate variable $\beta_t(s)$ we should:

1. Change direction of all transitions
2. Apply the same calculation logic as for $\alpha_t(s)$



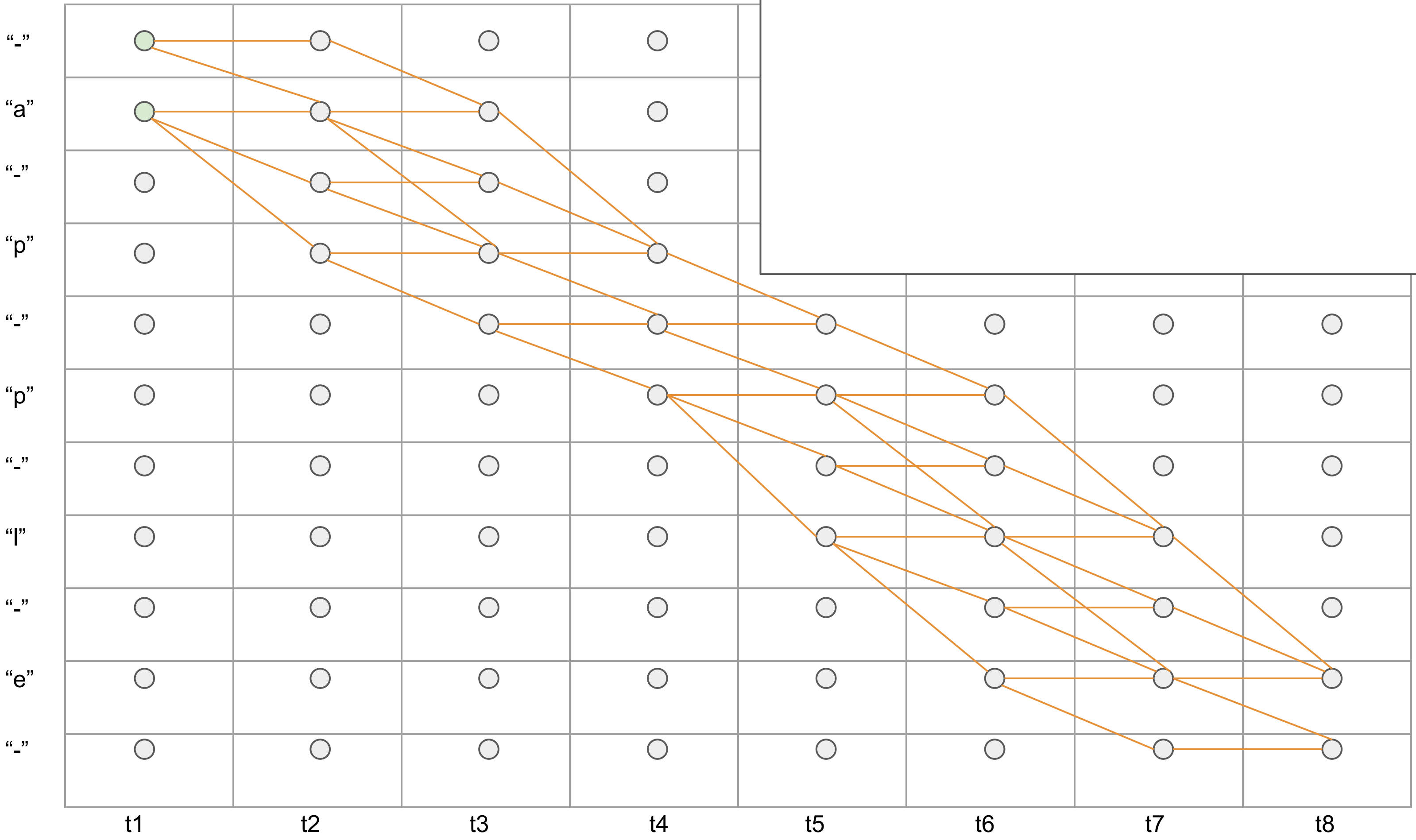


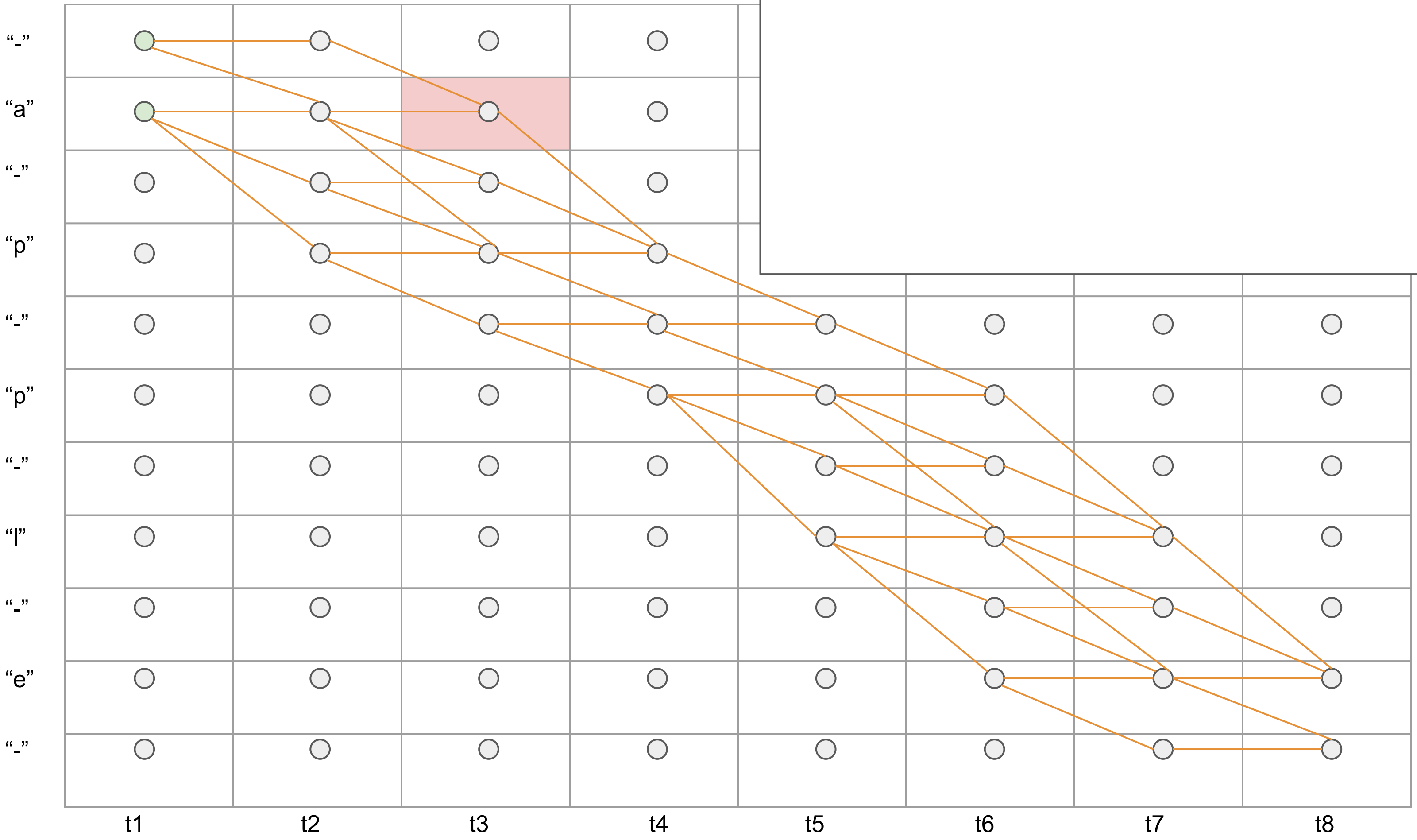
To calculate variable $\beta_t(s)$ we should:

1. Change direction of all transitions
2. Apply the same calculation logic as for $\alpha_t(s)$

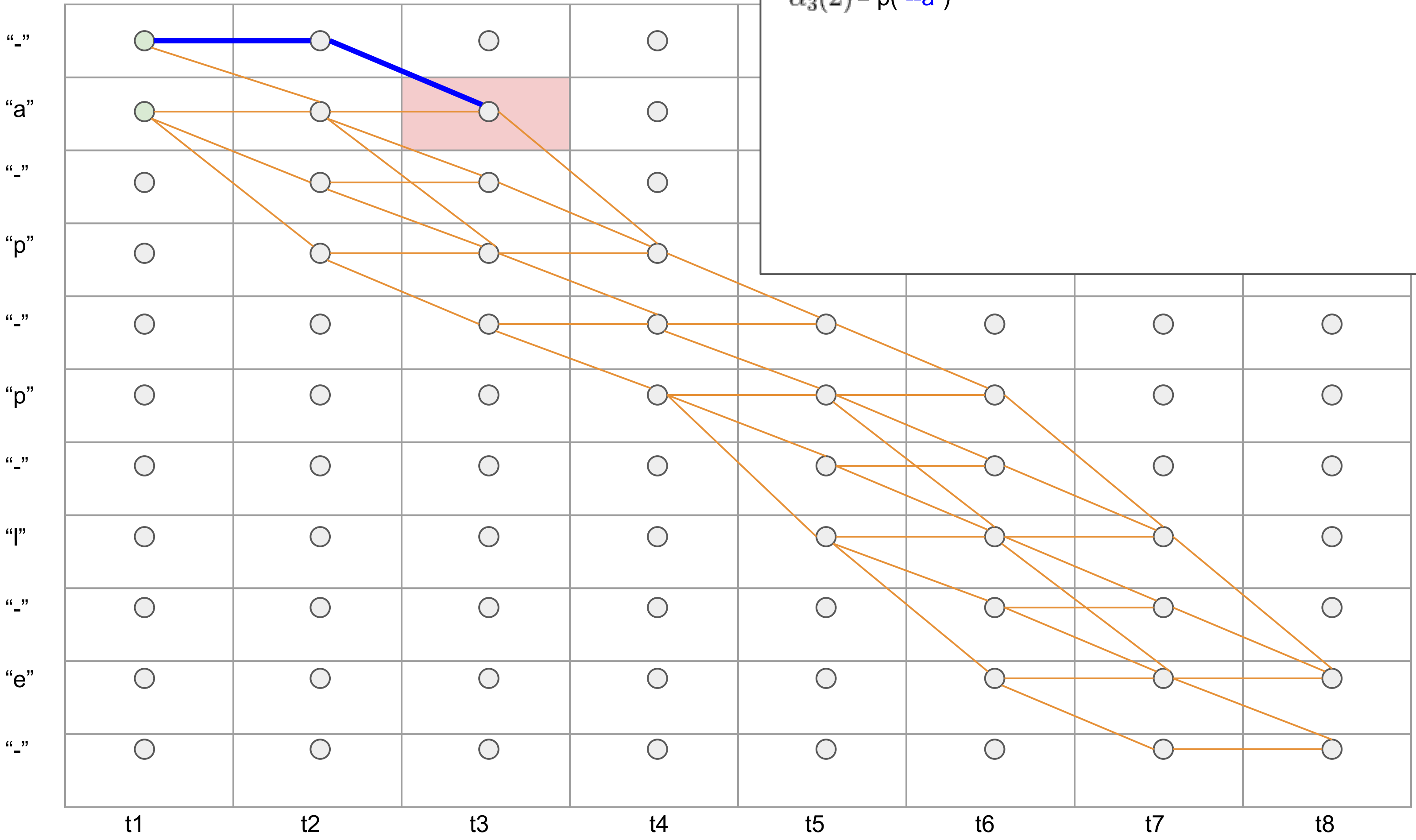
Note: start is here

I removed all directions. It is necessary for further reasoning.





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

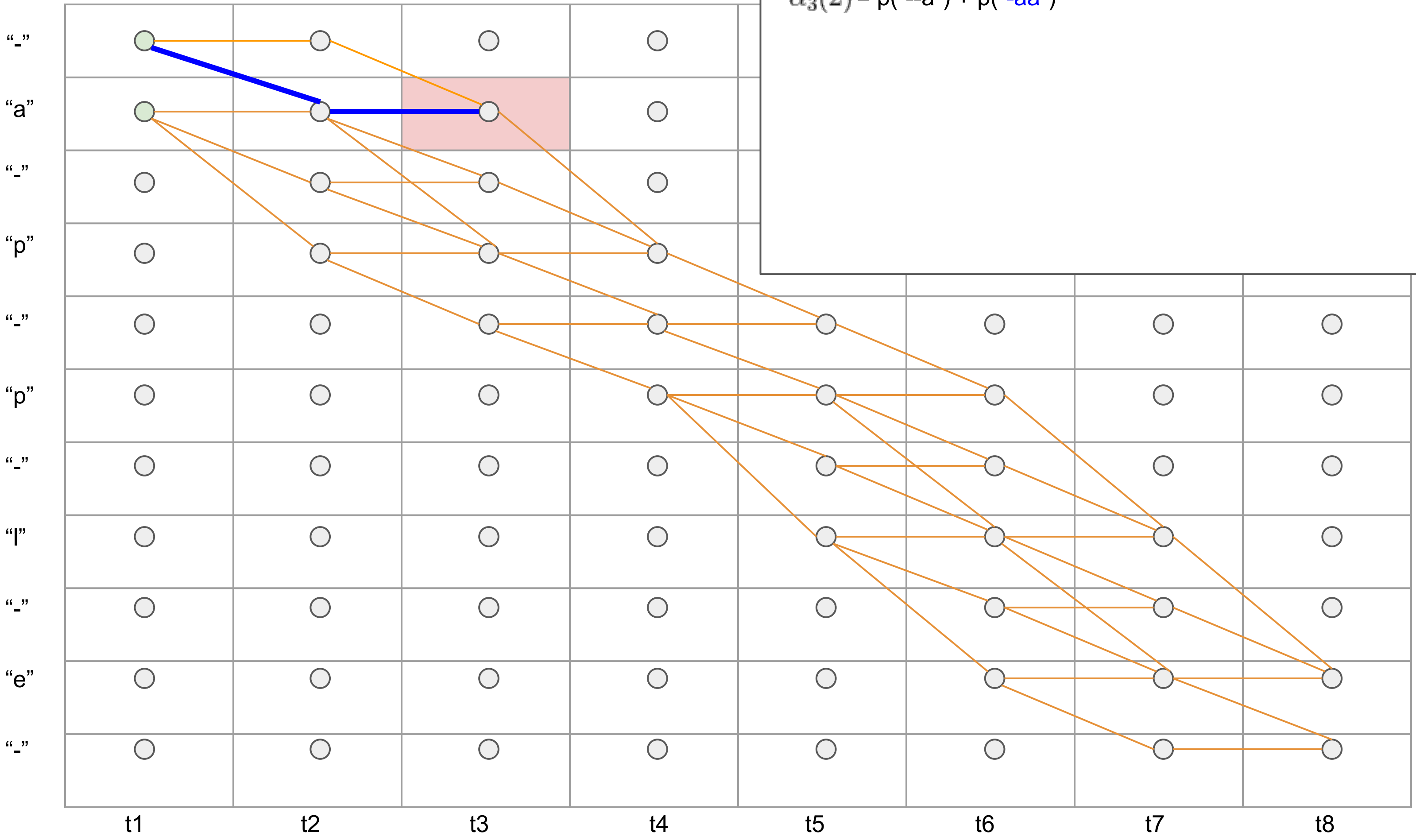


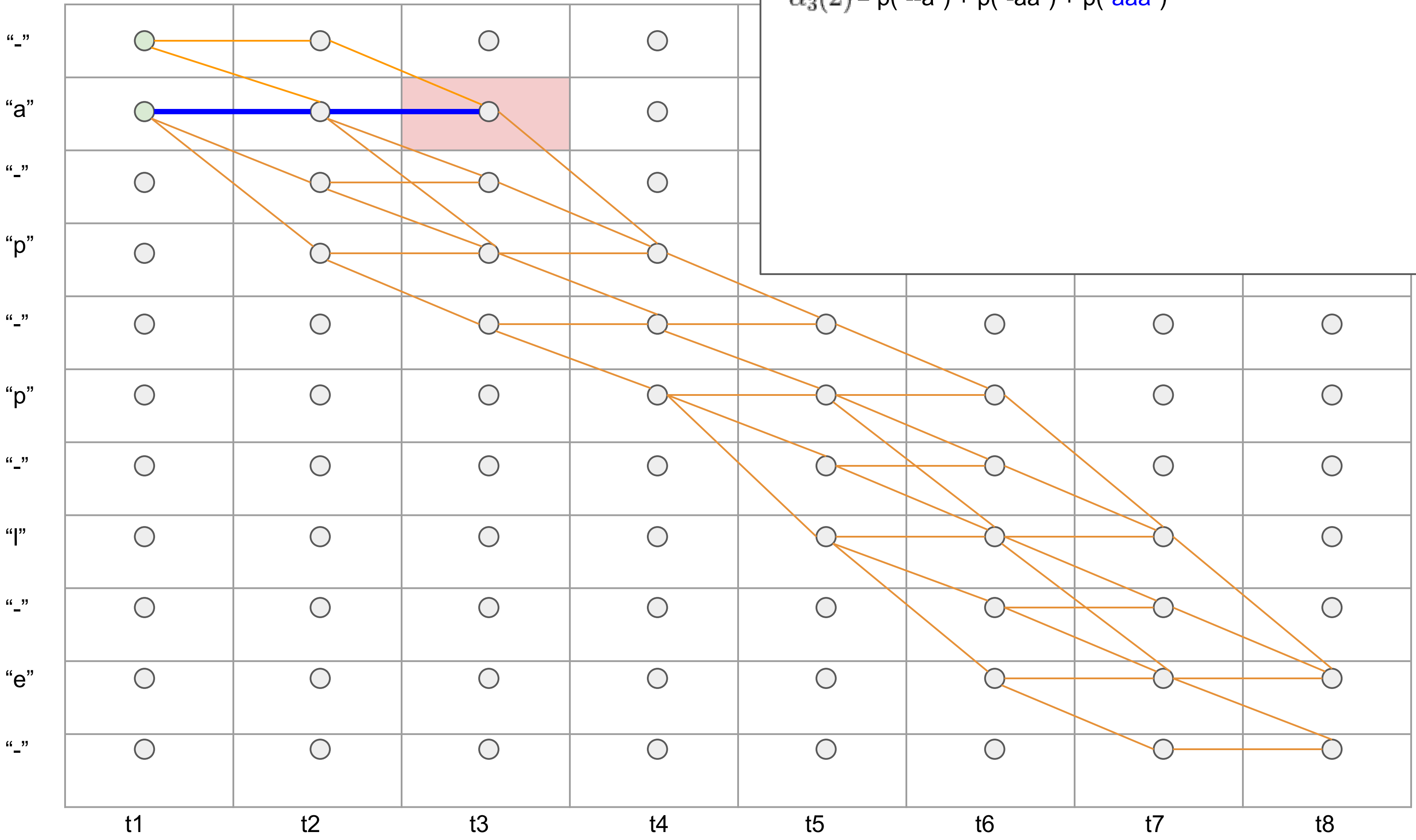
Let's consider t=3,s=2.

$\alpha_3(2)$ = p("--a")

Let’s consider t=3,s=2.

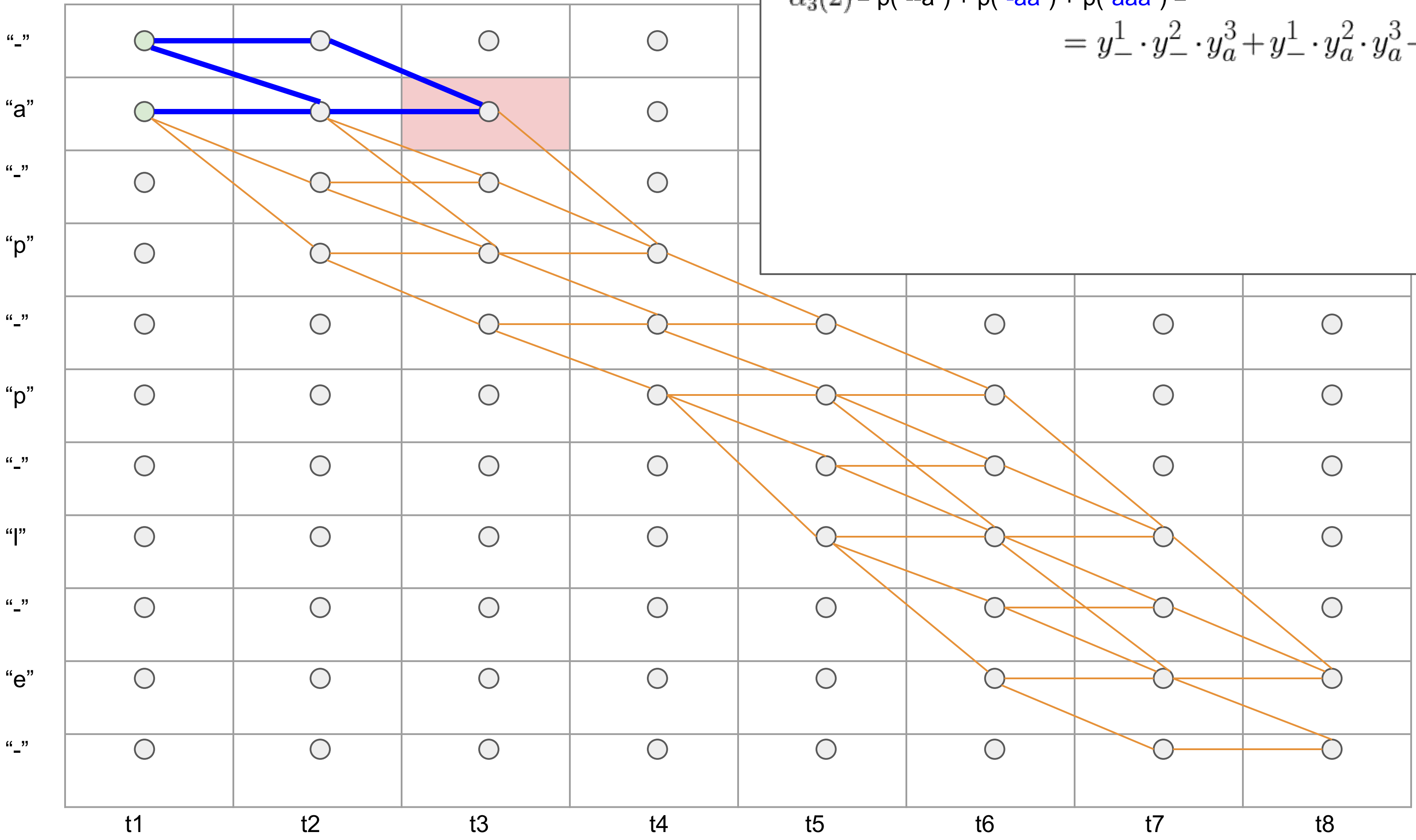
$\alpha_3(2)$ = p(“--a”) + p(“-aa”)





Let’s consider t=3,s=2.

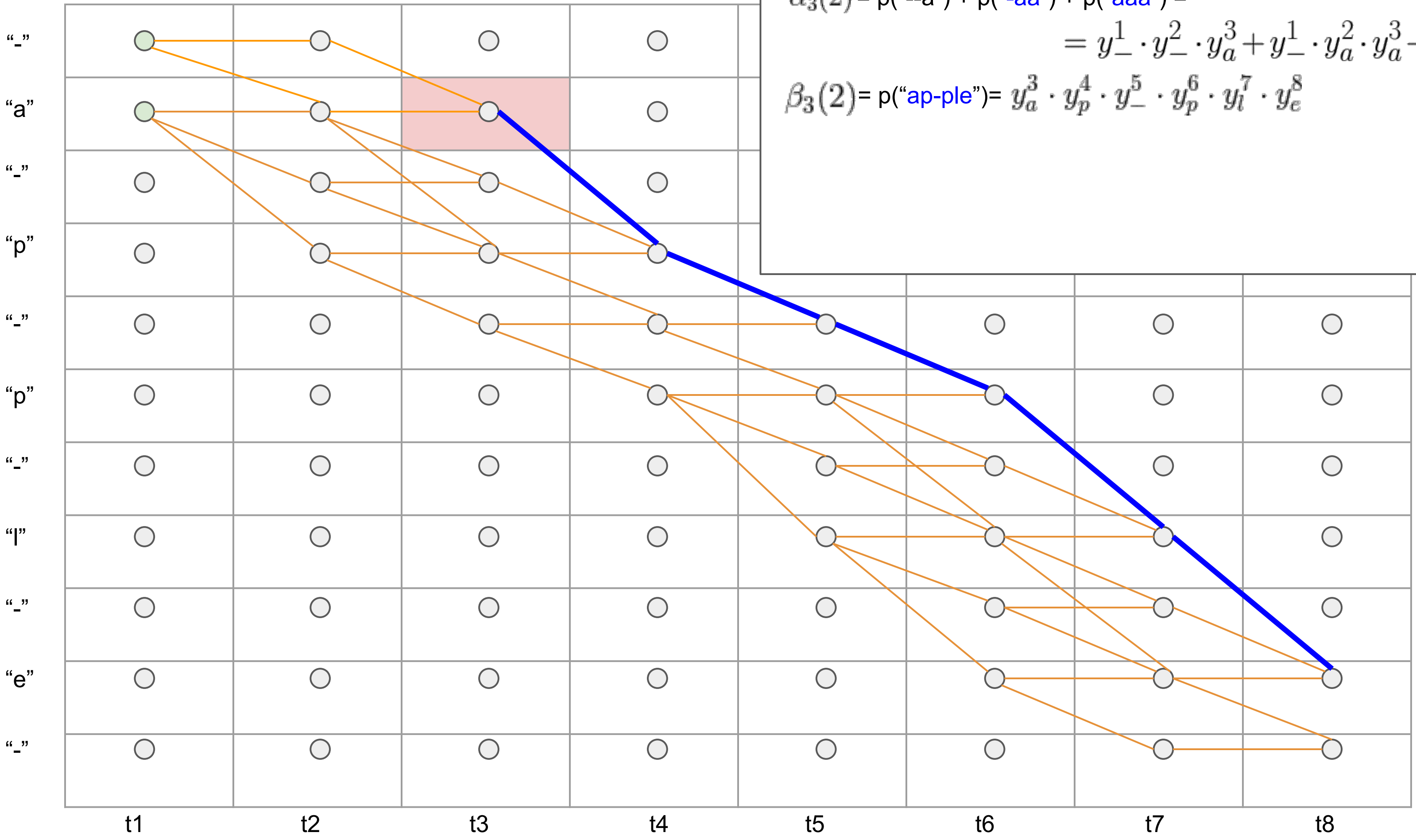
$\alpha_3(2)$ = p(“--a”) + p(“-aa”) + p(“aaa”)



Let’s consider t=3,s=2.

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

$$= y^1_- \cdot y^2_- \cdot y^3_a + y^1_- \cdot y^2_a \cdot y^3_a + y^1_a \cdot y^2_a \cdot y^3_a$$



Let’s consider t=3,s=2.

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

$$= y^1_- \cdot y^2_- \cdot y^3_a + y^1_- \cdot y^2_a \cdot y^3_a + y^1_a \cdot y^2_a \cdot y^3_a$$

$\beta_3(2)$= p(“ap-ple”)= $y^3_a \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$

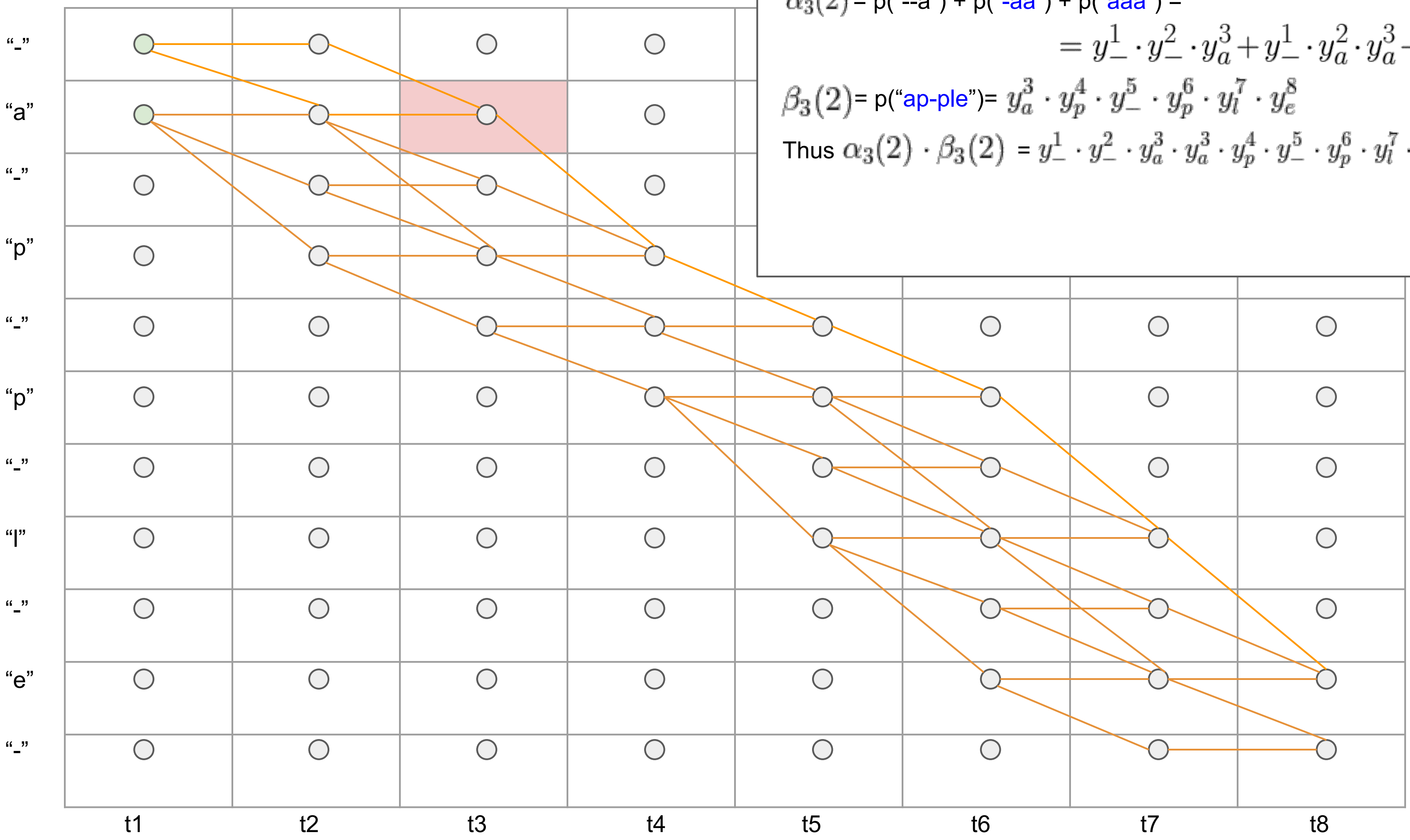
Let’s consider t=3,s=2.

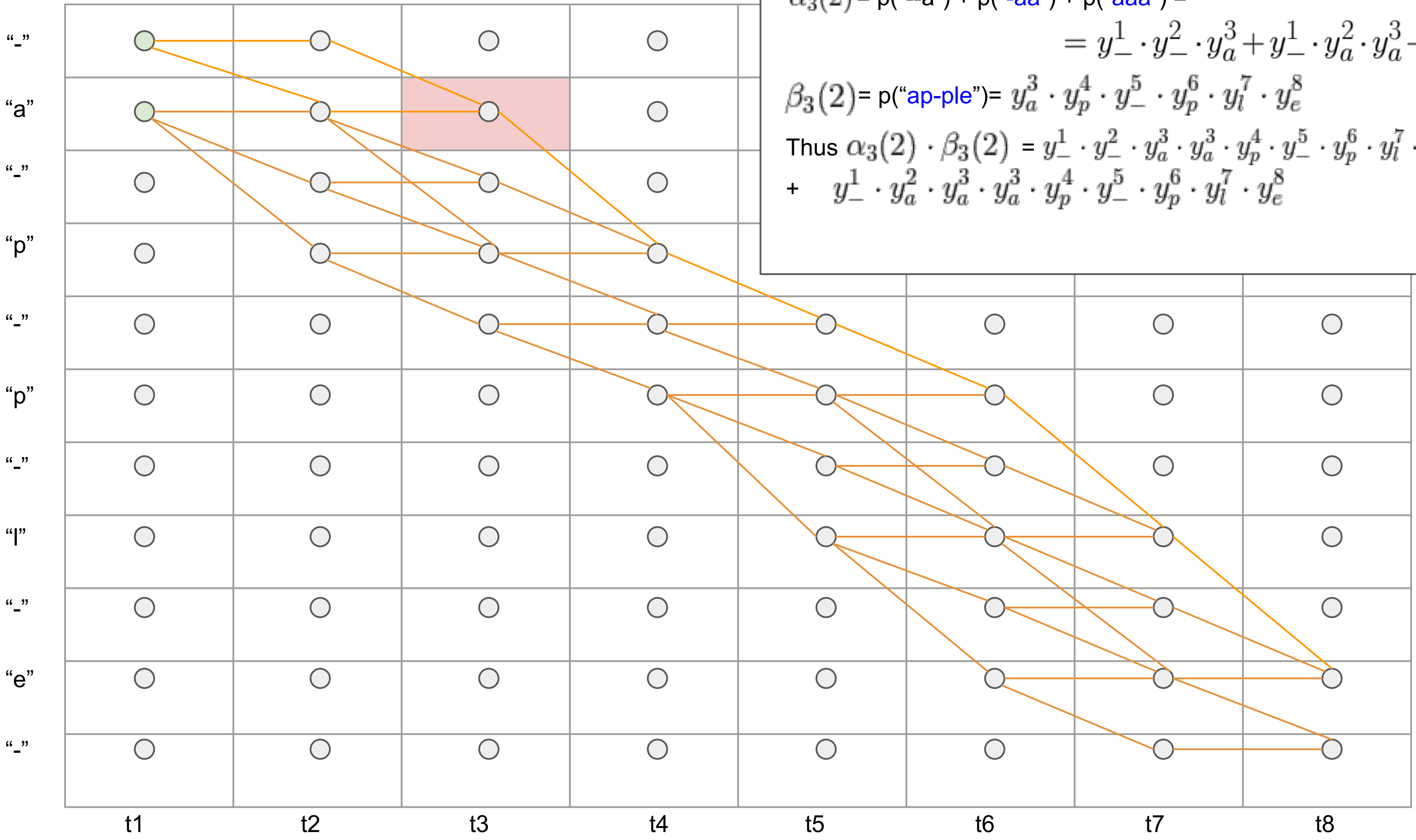
$\alpha_3(2)$ = p(“--a”) + p(“-aa”) + p(“aaa”) =

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

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





Let’s consider t=3,s=2.

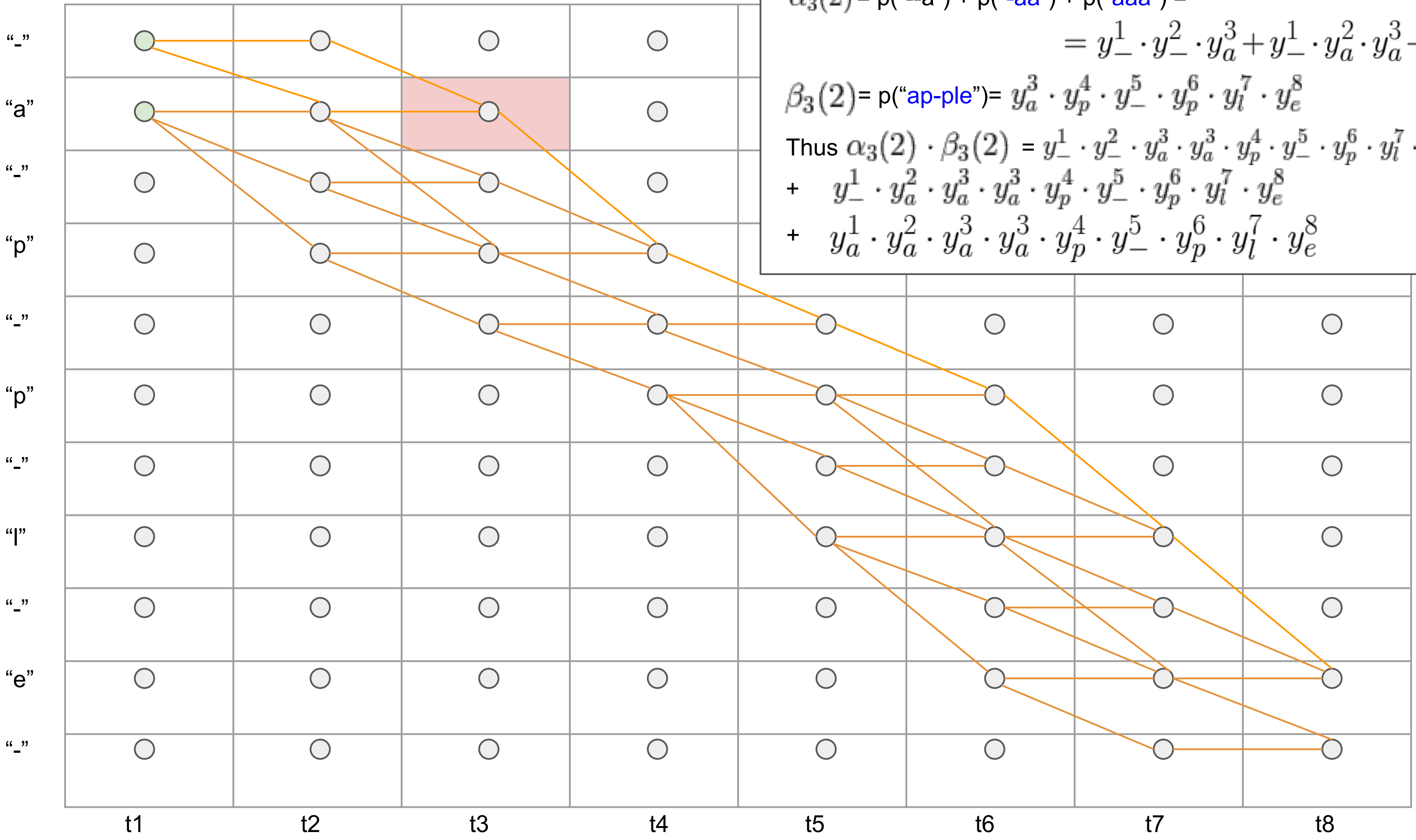
$\alpha_3(2)$ = p(“--a”) + p(“-aa”) + p(“aaa”) =

$$= y_{-}^{1} \cdot y_{-}^{2} \cdot y_{a}^{3} + y_{-}^{1} \cdot y_{a}^{2} \cdot y_{a}^{3} + y_{a}^{1} \cdot y_{a}^{2} \cdot y_{a}^{3}$$

$\beta_3(2)$= 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 $\alpha_3(2) \cdot \beta_3(2)$ = $y_{-}^{1} \cdot y_{-}^{2} \cdot y_{a}^{3} \cdot y_{a}^{3} \cdot y_{p}^{4} \cdot y_{-}^{5} \cdot y_{p}^{6} \cdot y_{l}^{7} \cdot y_{e}^{8}$ +

+ $y_{-}^{1} \cdot y_{a}^{2} \cdot y_{a}^{3} \cdot y_{a}^{3} \cdot y_{p}^{4} \cdot y_{-}^{5} \cdot y_{p}^{6} \cdot y_{l}^{7} \cdot y_{e}^{8}$



Let’s consider t=3,s=2.

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

$$= y^1_- \cdot y^2_- \cdot y^3_a + y^1_- \cdot y^2_a \cdot y^3_a + y^1_a \cdot y^2_a \cdot y^3_a$$

$\beta_3(2)$= p(“ap-ple”)= $y^3_a \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$

Thus $\alpha_3(2) \cdot \beta_3(2)$ = $y^1_- \cdot y^2_- \cdot y^3_a \cdot y^3_a \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$ +

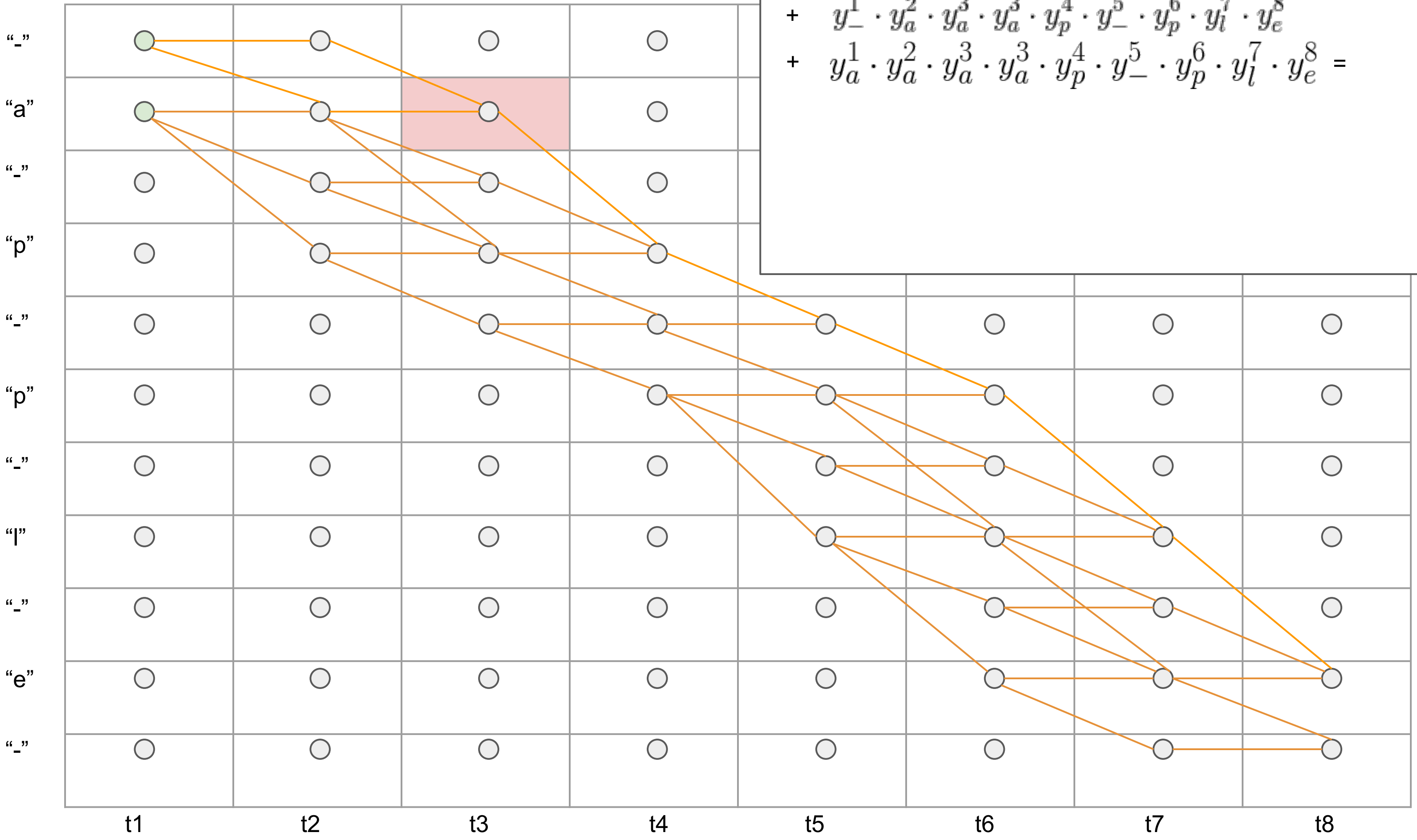
+ $y^1_- \cdot y^2_a \cdot y^3_a \cdot y^3_a \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$

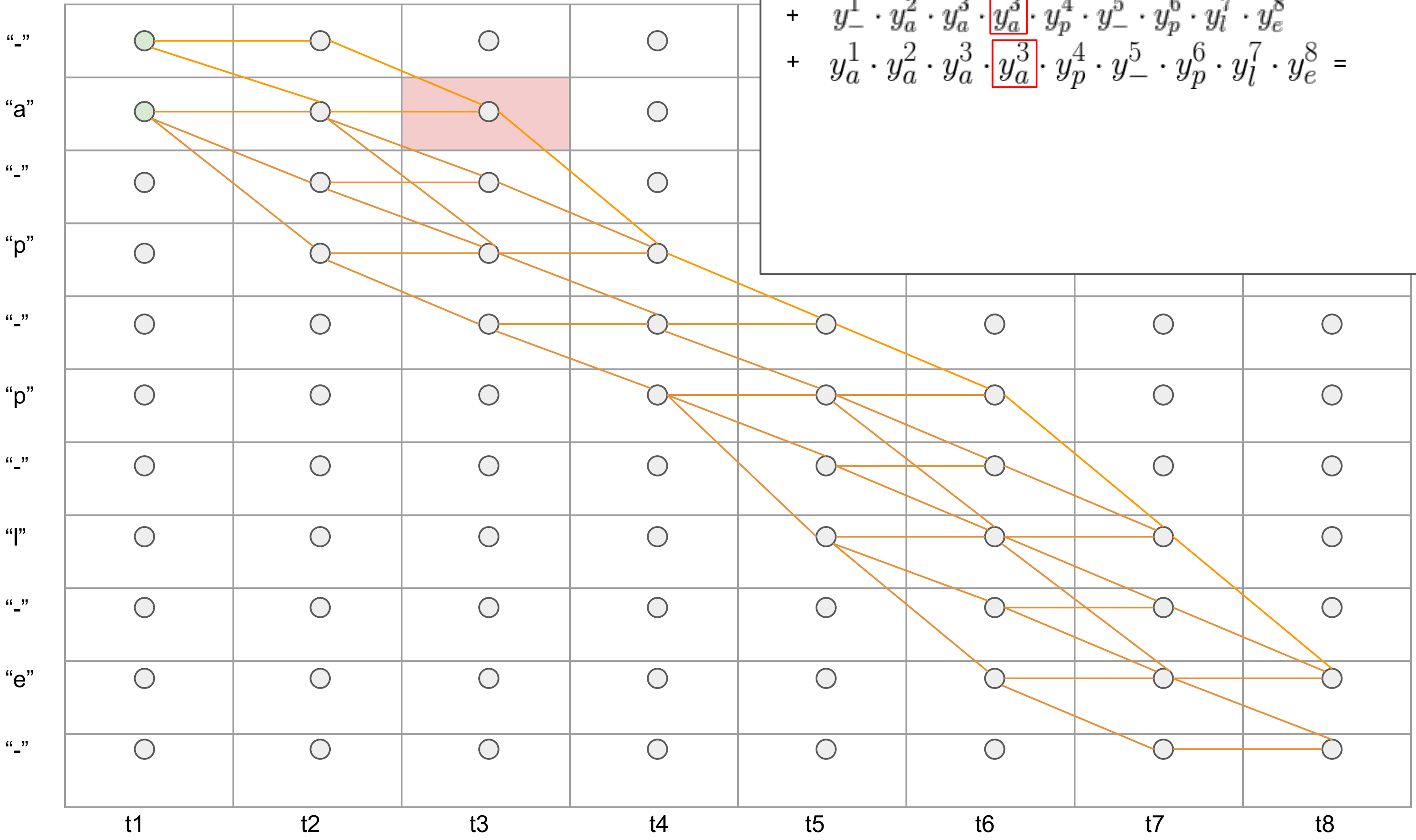
+ $y^1_a \cdot y^2_a \cdot y^3_a \cdot y^3_a \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$

Thus $\alpha_3(2) \cdot \beta_3(2)$ = $y^1_- \cdot y^2_- \cdot y^3_a \cdot y^3_a \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$ +

+ $y^1_- \cdot y^2_a \cdot y^3_a \cdot y^3_a \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$

+ $y^1_a \cdot y^2_a \cdot y^3_a \cdot y^3_a \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$ =

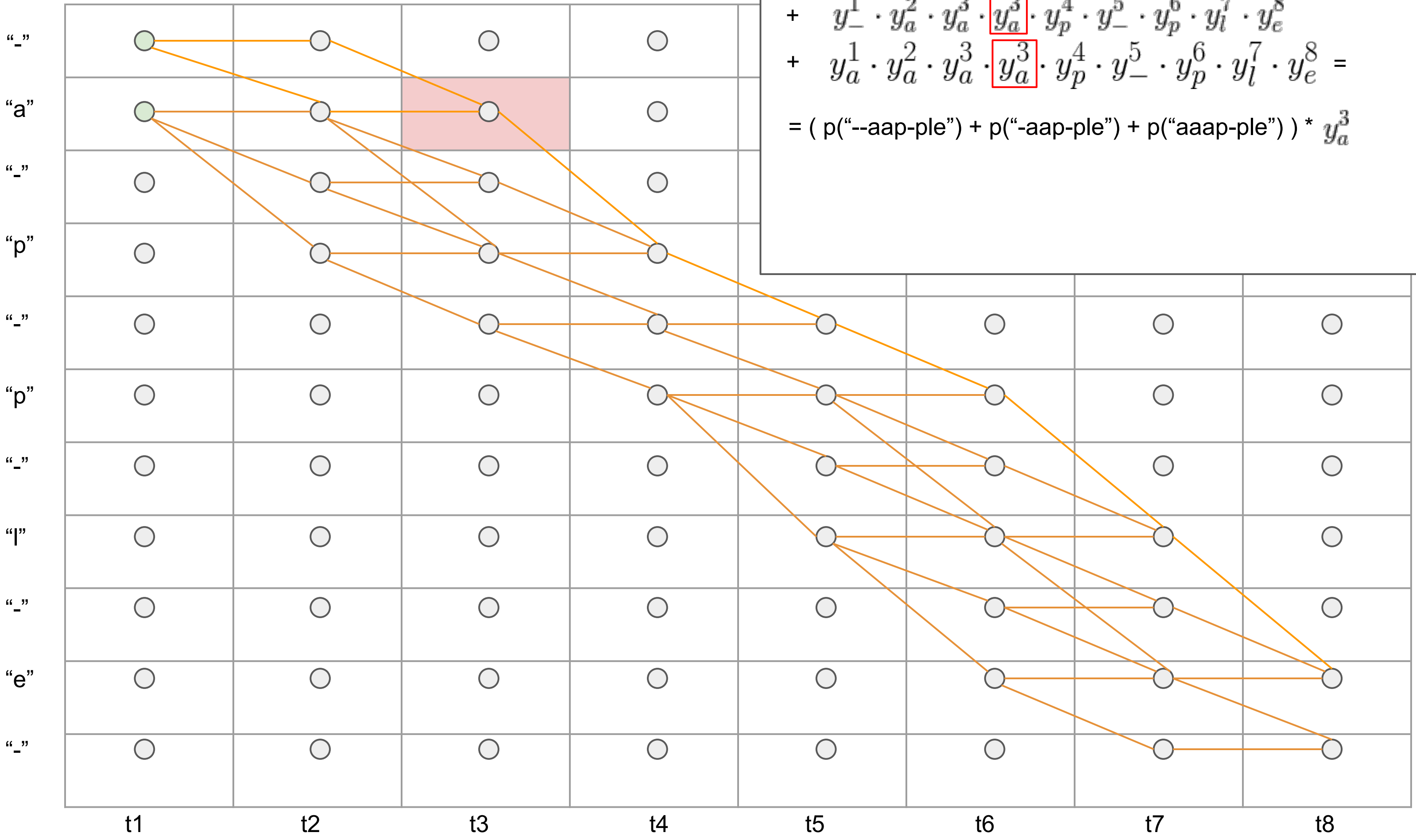




Thus $\alpha_3(2) \cdot \beta_3(2)$ = $y^1_- \cdot y^2_- \cdot y^3_a \cdot \boxed{y^3_a} \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$ +

+ $y^1_- \cdot y^2_a \cdot y^3_a \cdot \boxed{y^3_a} \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$

+ $y^1_a \cdot y^2_a \cdot y^3_a \cdot \boxed{y^3_a} \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$ =



Thus $\alpha_3(2) \cdot \beta_3(2)$ = $y^1_- \cdot y^2_- \cdot y^3_a \cdot \boxed{y^3_a} \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$ +

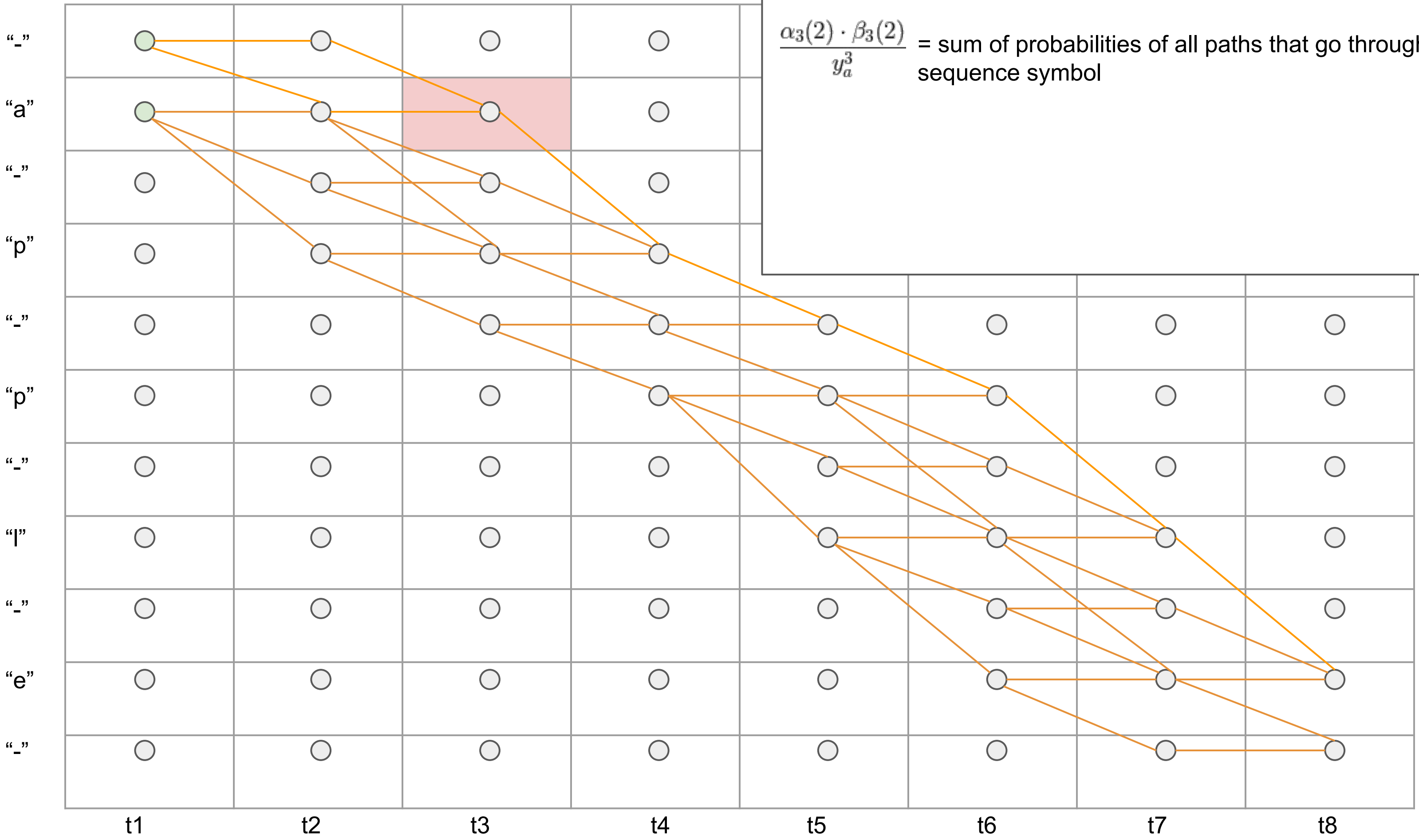
+ $y^1_- \cdot y^2_a \cdot y^3_a \cdot \boxed{y^3_a} \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$

+ $y^1_a \cdot y^2_a \cdot y^3_a \cdot \boxed{y^3_a} \cdot y^4_p \cdot y^5_- \cdot y^6_p \cdot y^7_l \cdot y^8_e$ =

= ( p(“--aap-ple”) + p(“-aap-ple”) + p(“aaap-ple”) ) * $y^3_a$

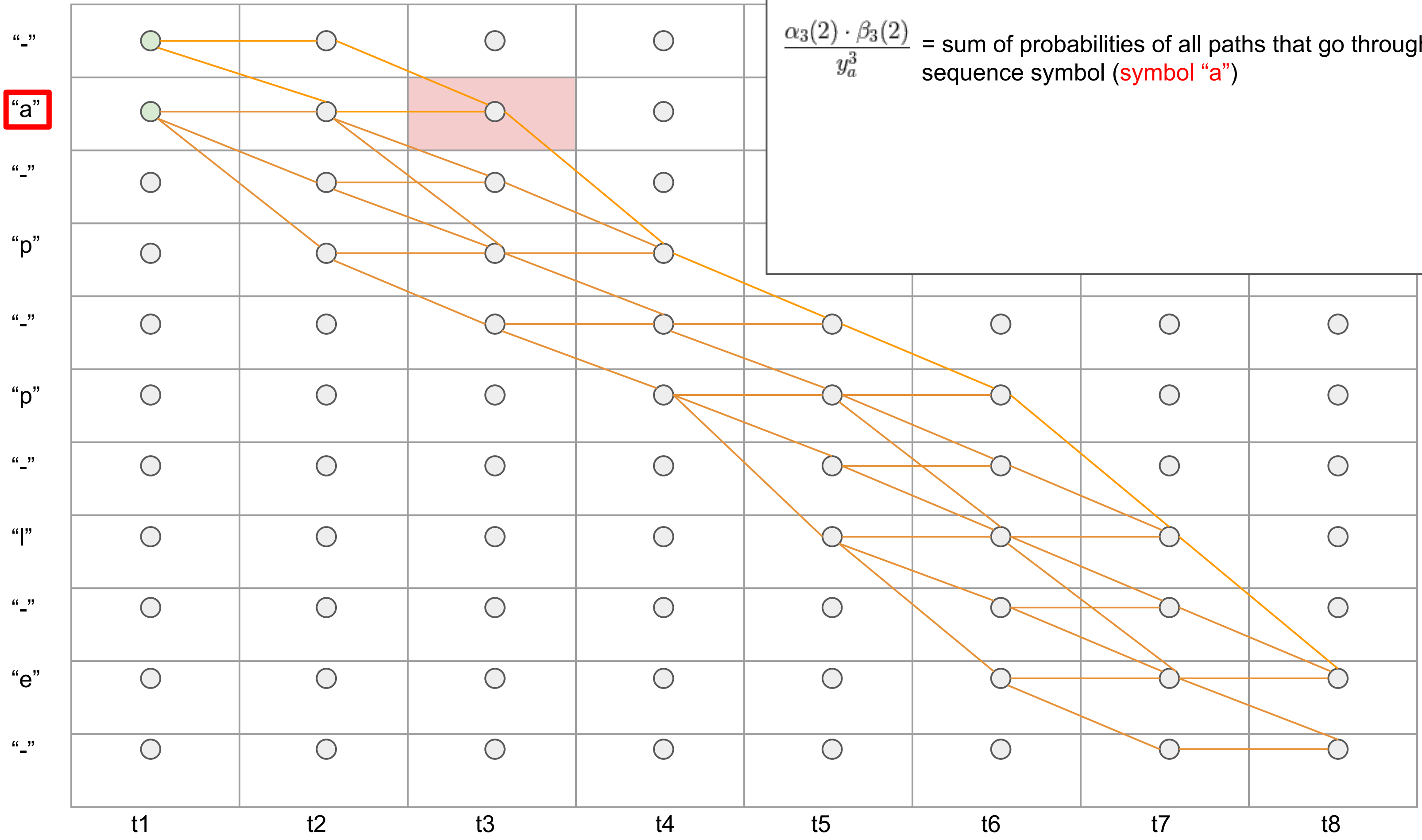
$\alpha_3(2) \cdot \beta_3(2)$ = ( p(“--aap-ple”) + p(“-aap-ple”) + p(“aaap-ple”) ) * $y_a^3$

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



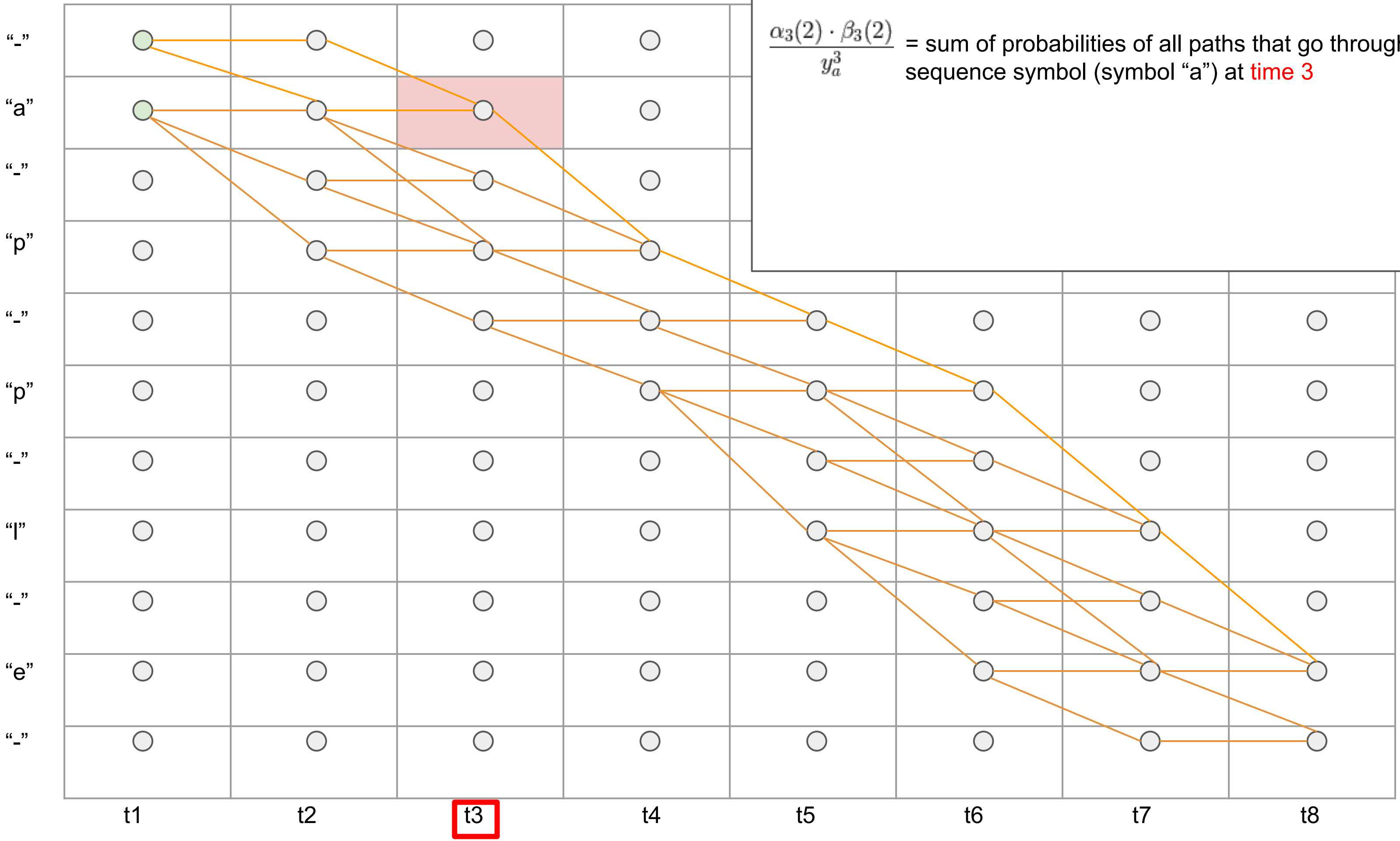
$\alpha_3(2) \cdot \beta_3(2)$ = ( p(“--aap-ple”) + p(“-aap-ple”) + p(“aaap-ple”) ) * $y_a^3$

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



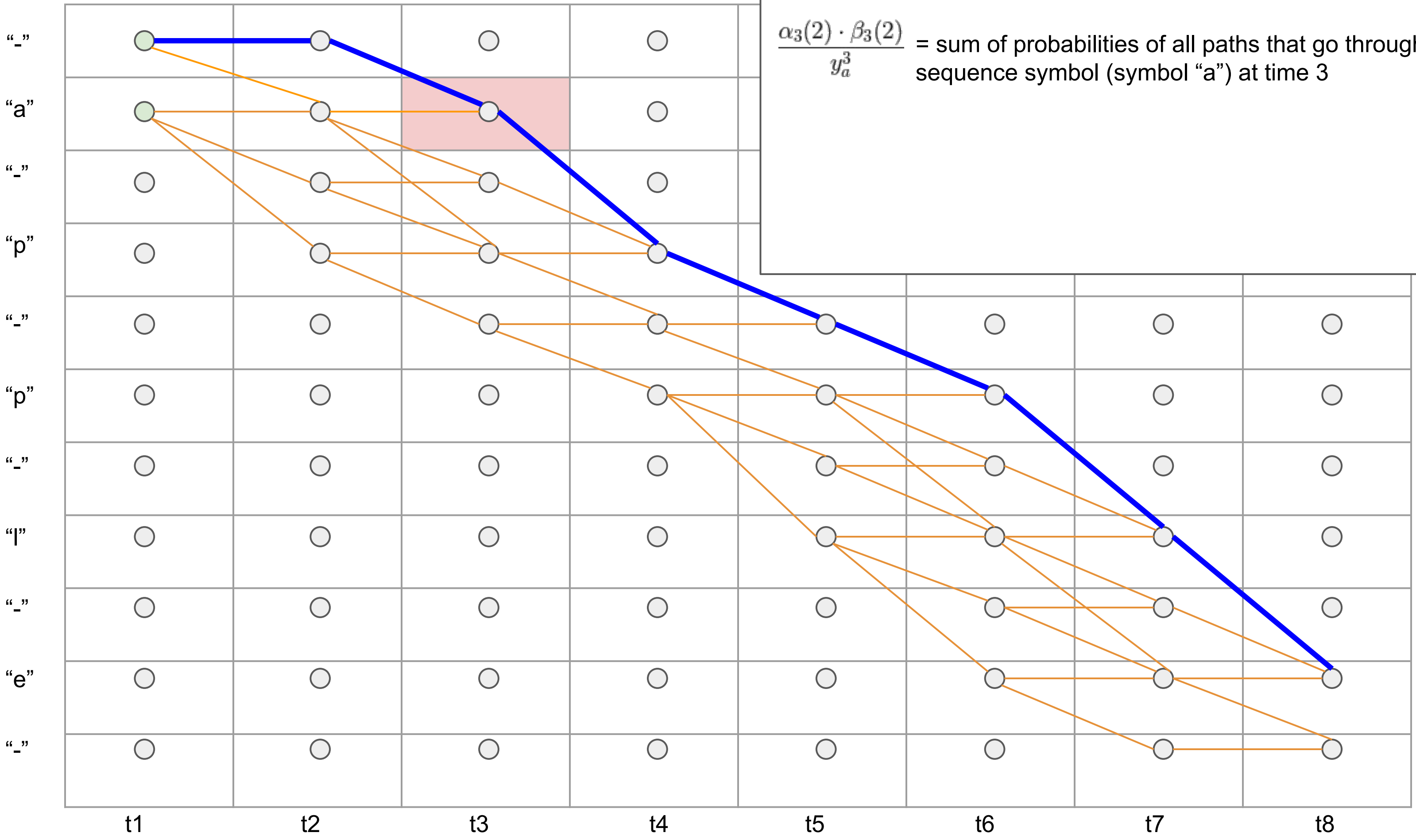
$\alpha_3(2) \cdot \beta_3(2)$ = ( p(“--aap-ple”) + p(“-aap-ple”) + p(“aaap-ple”) ) * $y_a^3$

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



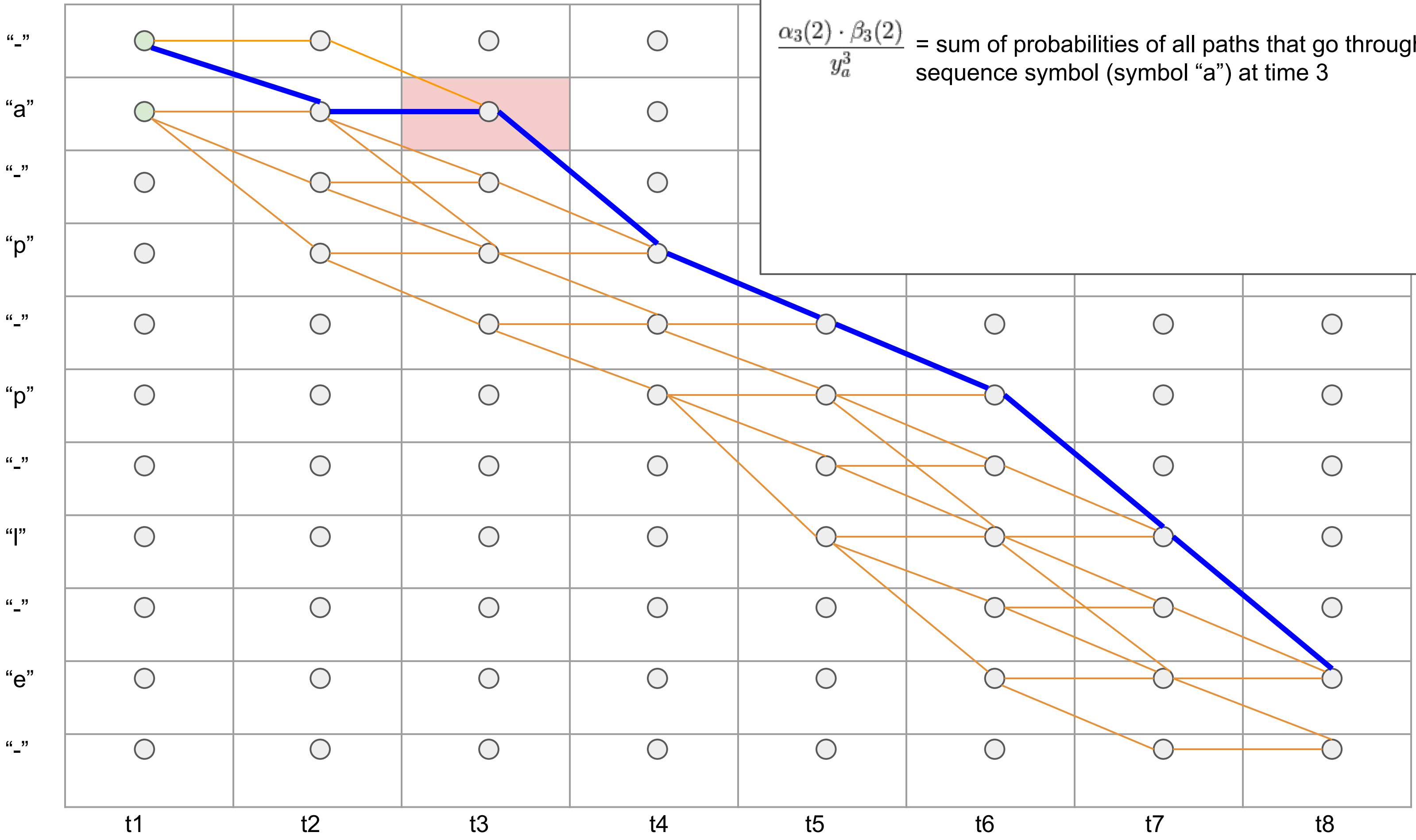
$\alpha_3(2) \cdot \beta_3(2)$= ( p(“--aap-ple”) + p(“-aap-ple”) + p(“aaap-ple”) ) * $y_a^3$

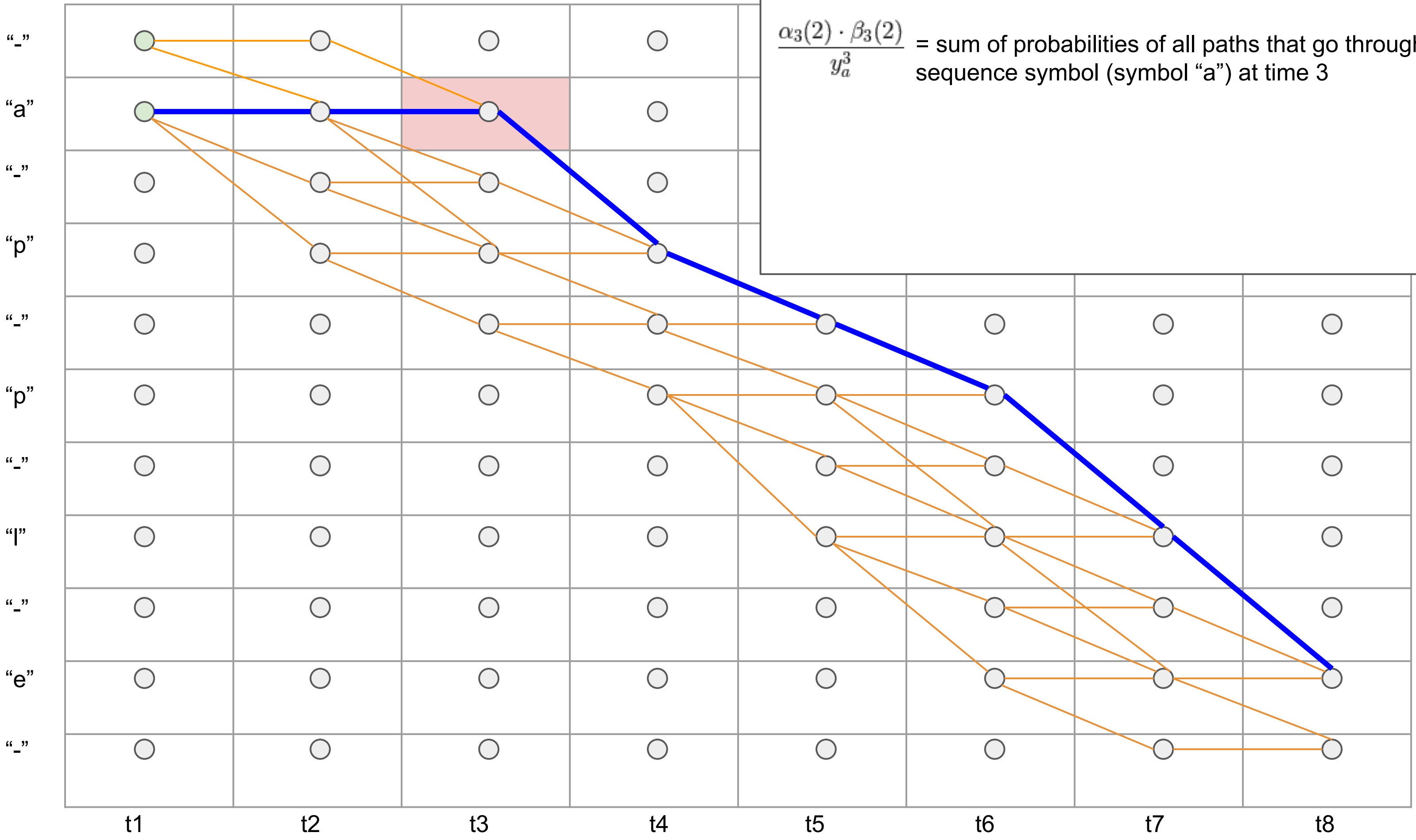
$\frac{\alpha_3(2) \cdot \beta_3(2)}{y_a^3}$ = sum of probabilities of all paths that go through second sequence symbol (symbol “a”) at time 3



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

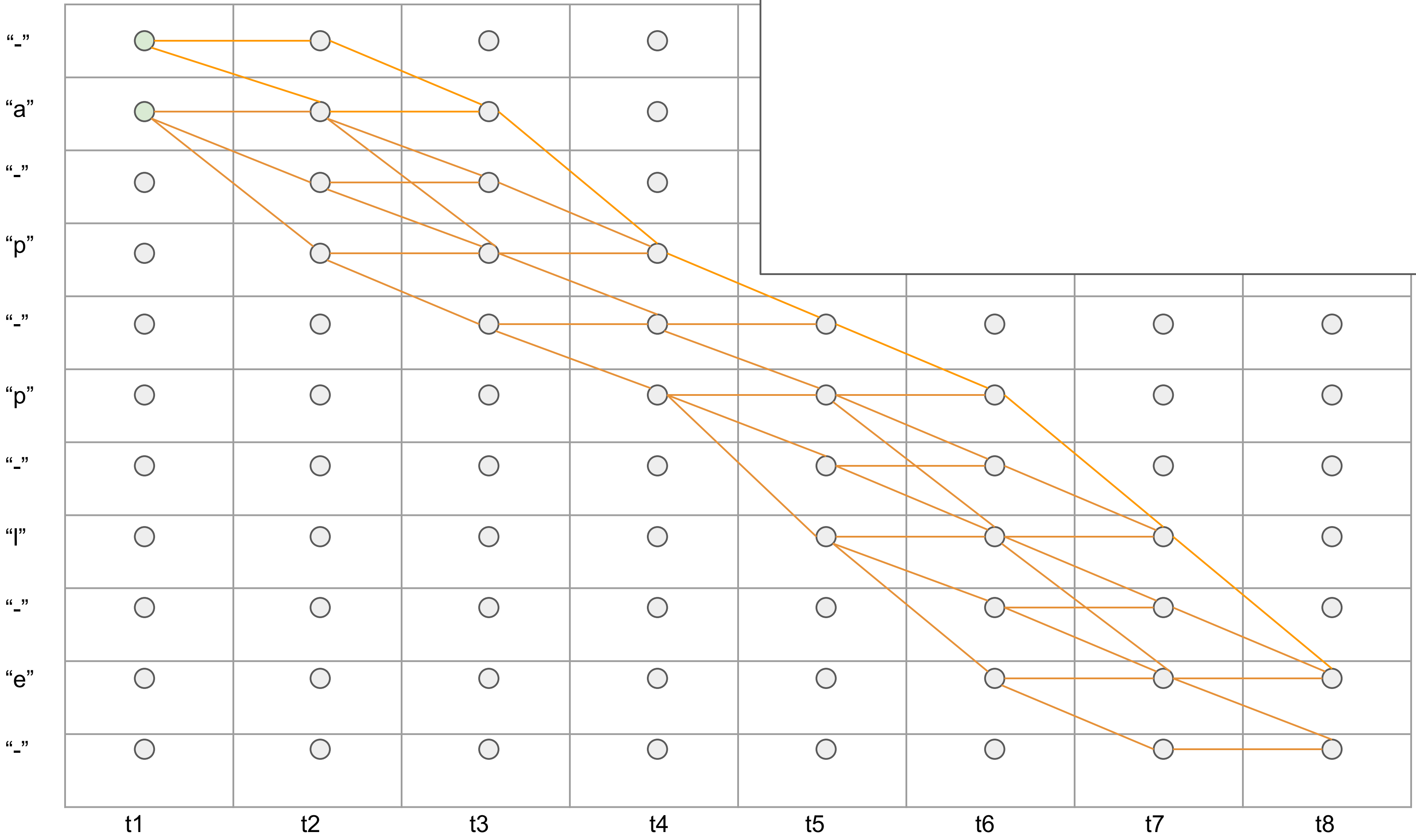
$\frac{\alpha_3(2) \cdot \beta_3(2)}{y_a^3}$ = sum of probabilities of all paths that go through second sequence symbol (symbol “a”) at time 3





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

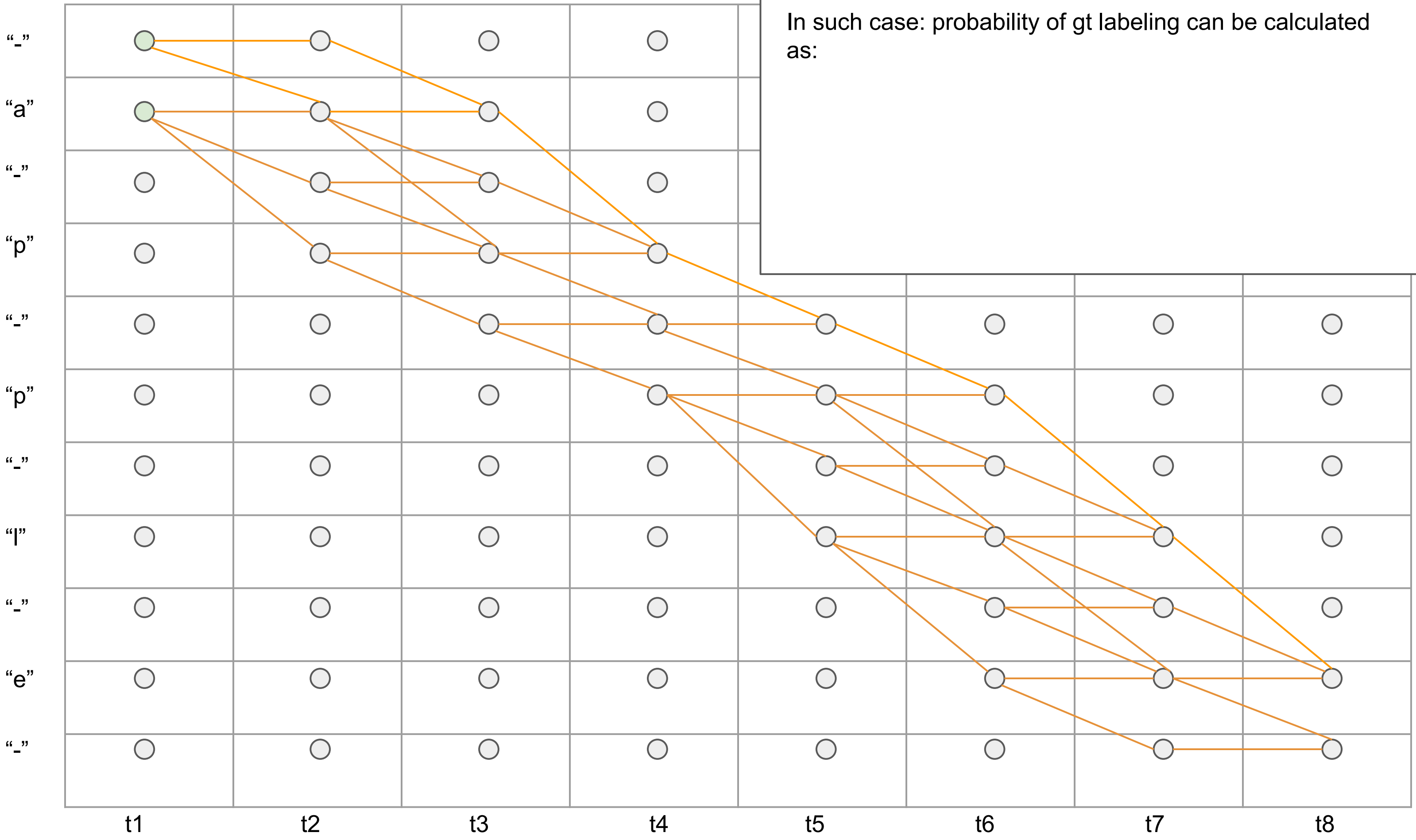
$\frac{\alpha_3(2) \cdot \beta_3(2)}{y_a^3}$ = sum of probabilities of all paths that go through second sequence symbol (symbol "a") at time 3



Let’s commit: t=3

Let’s commit: t=3

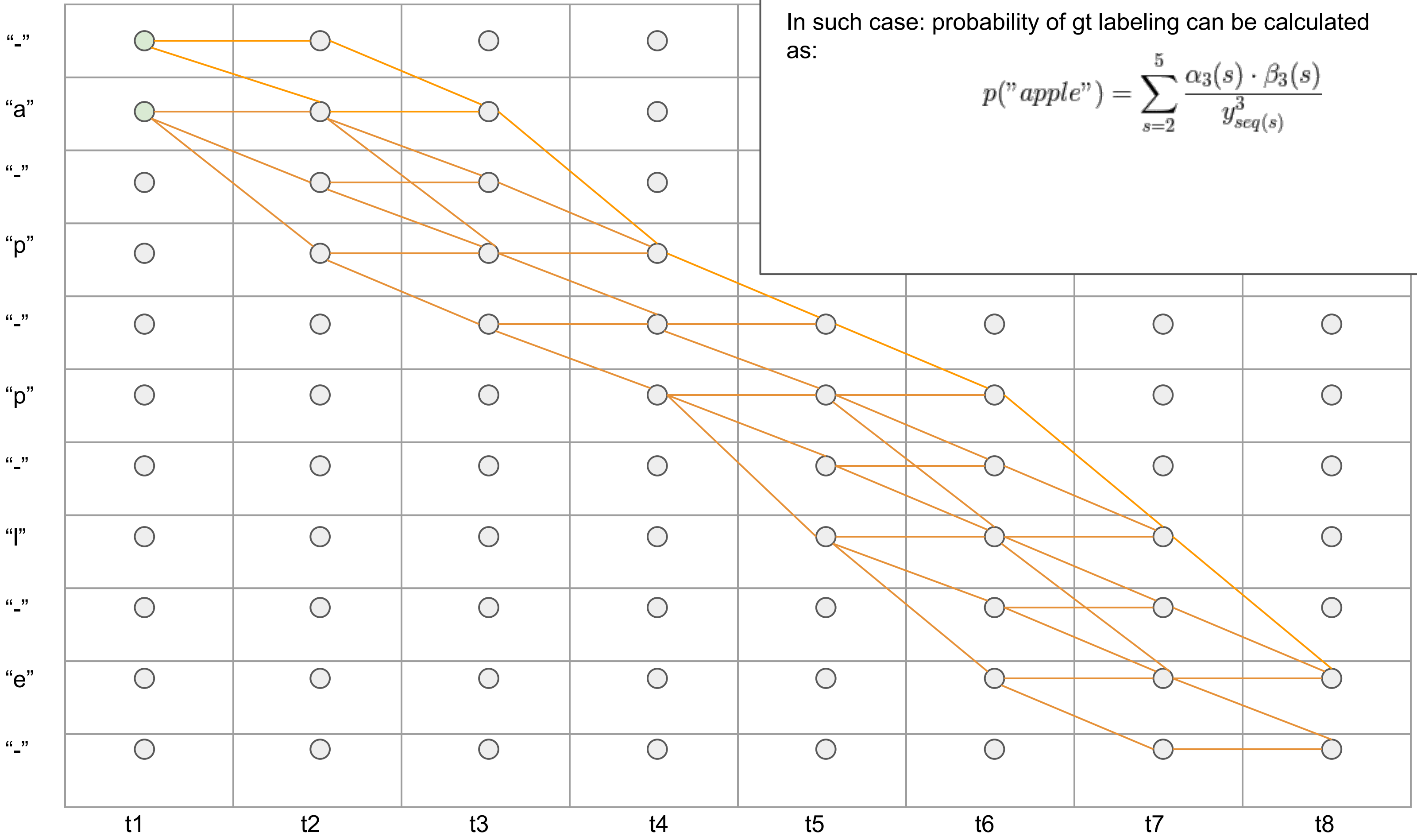
In such case: probability of gt labeling can be calculated as:

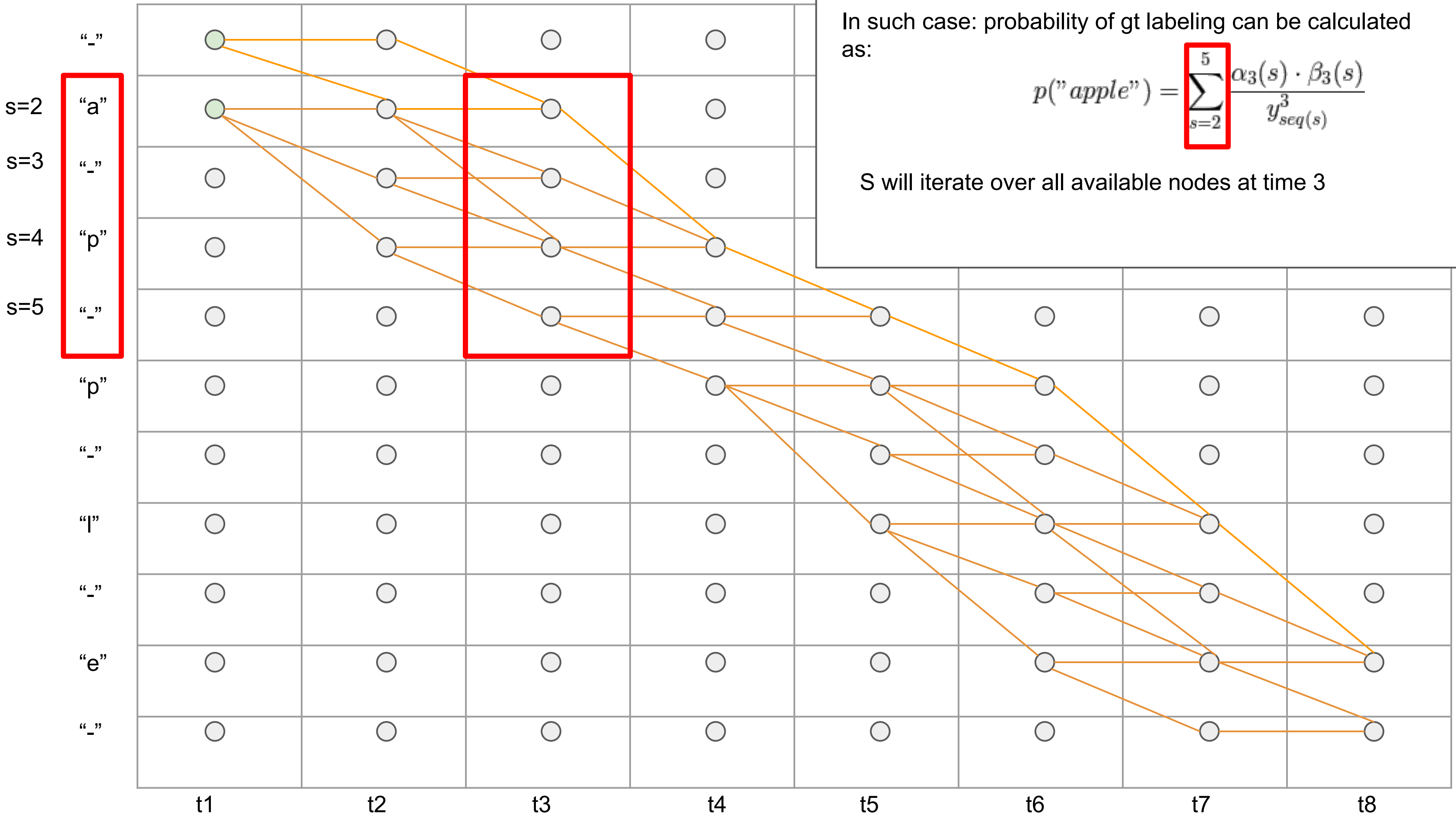


Let’s commit: t=3

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

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





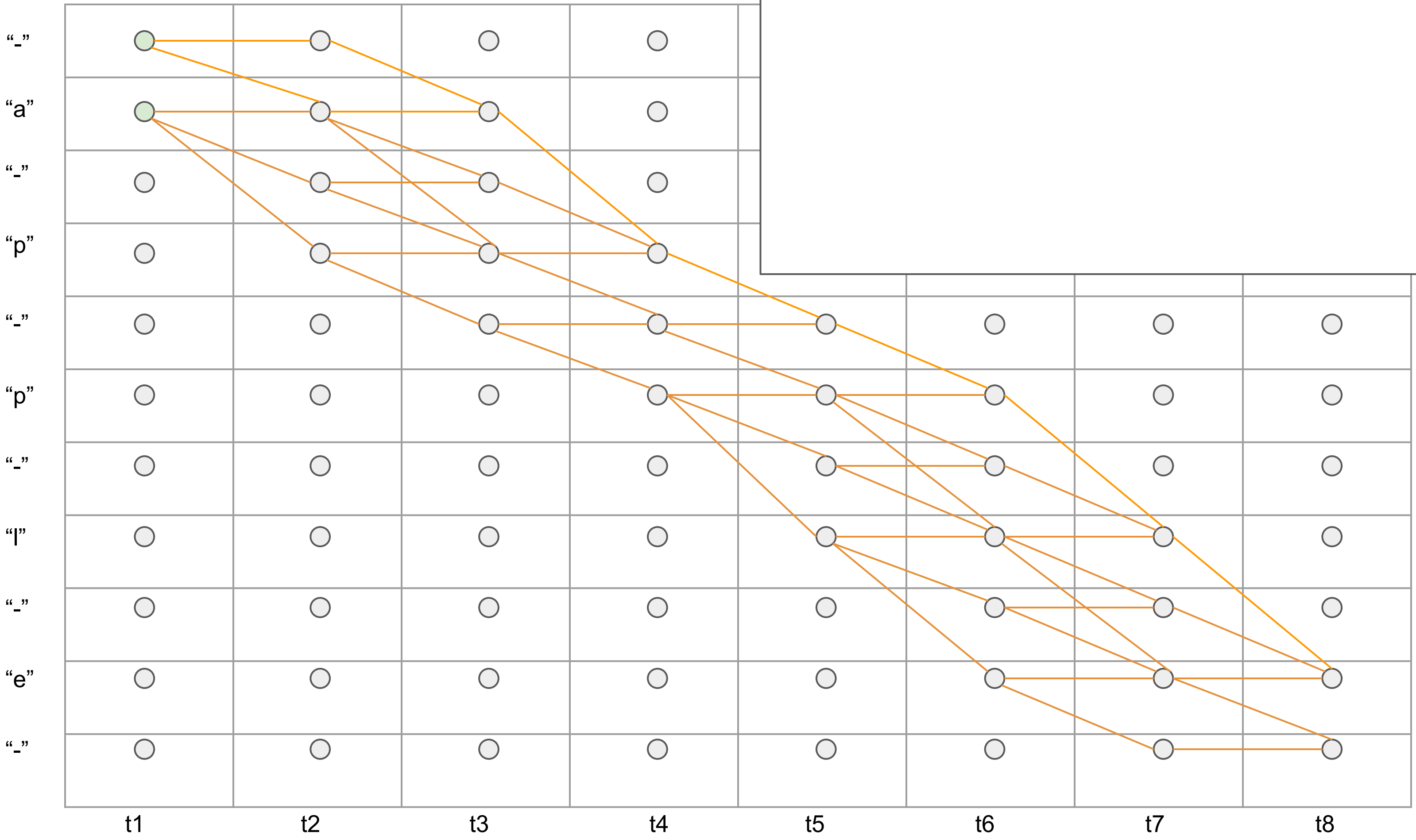
Let’s commit: t=3

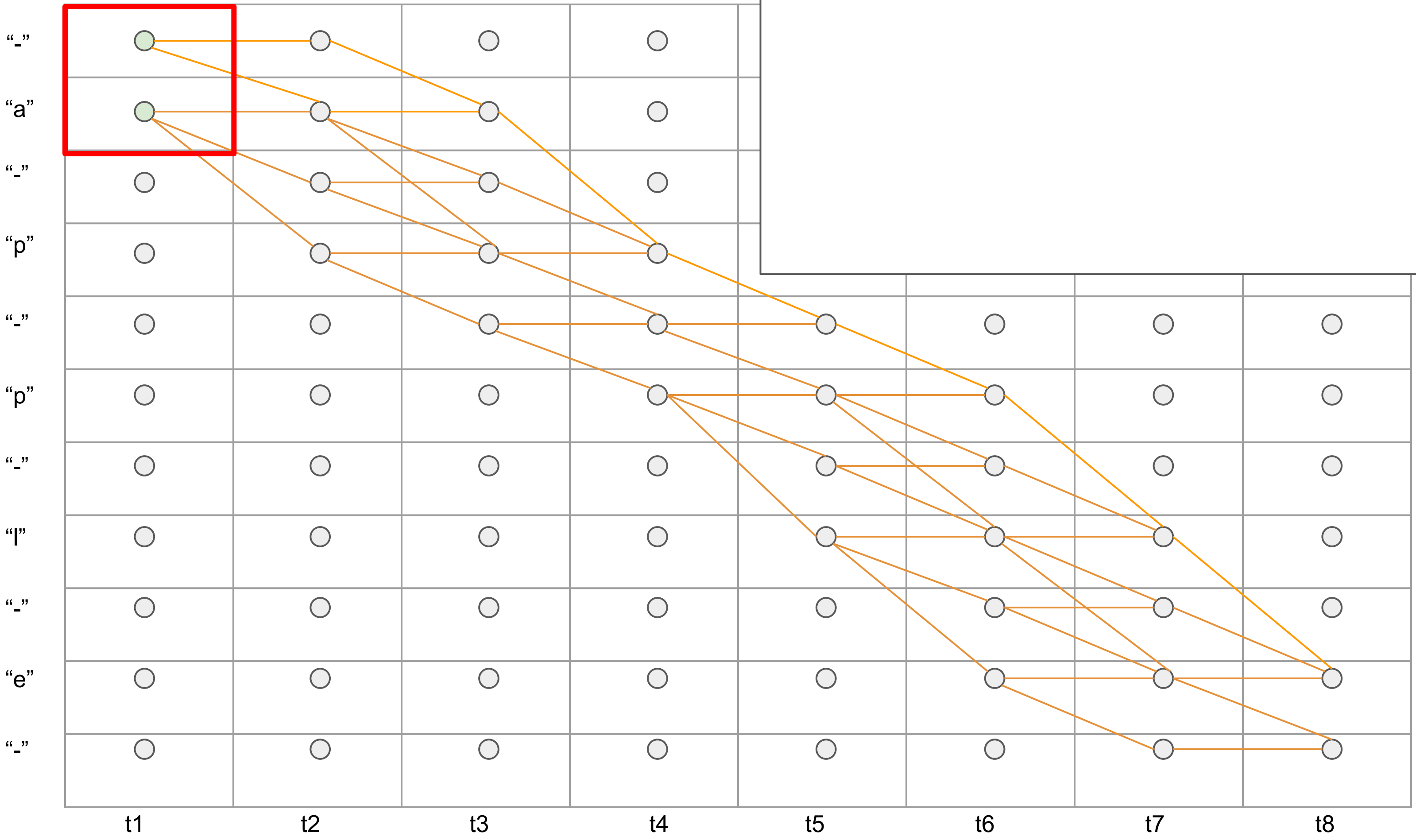
In such case: probability of gt labeling can be calculated as:

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

S will iterate over all available nodes at time 3

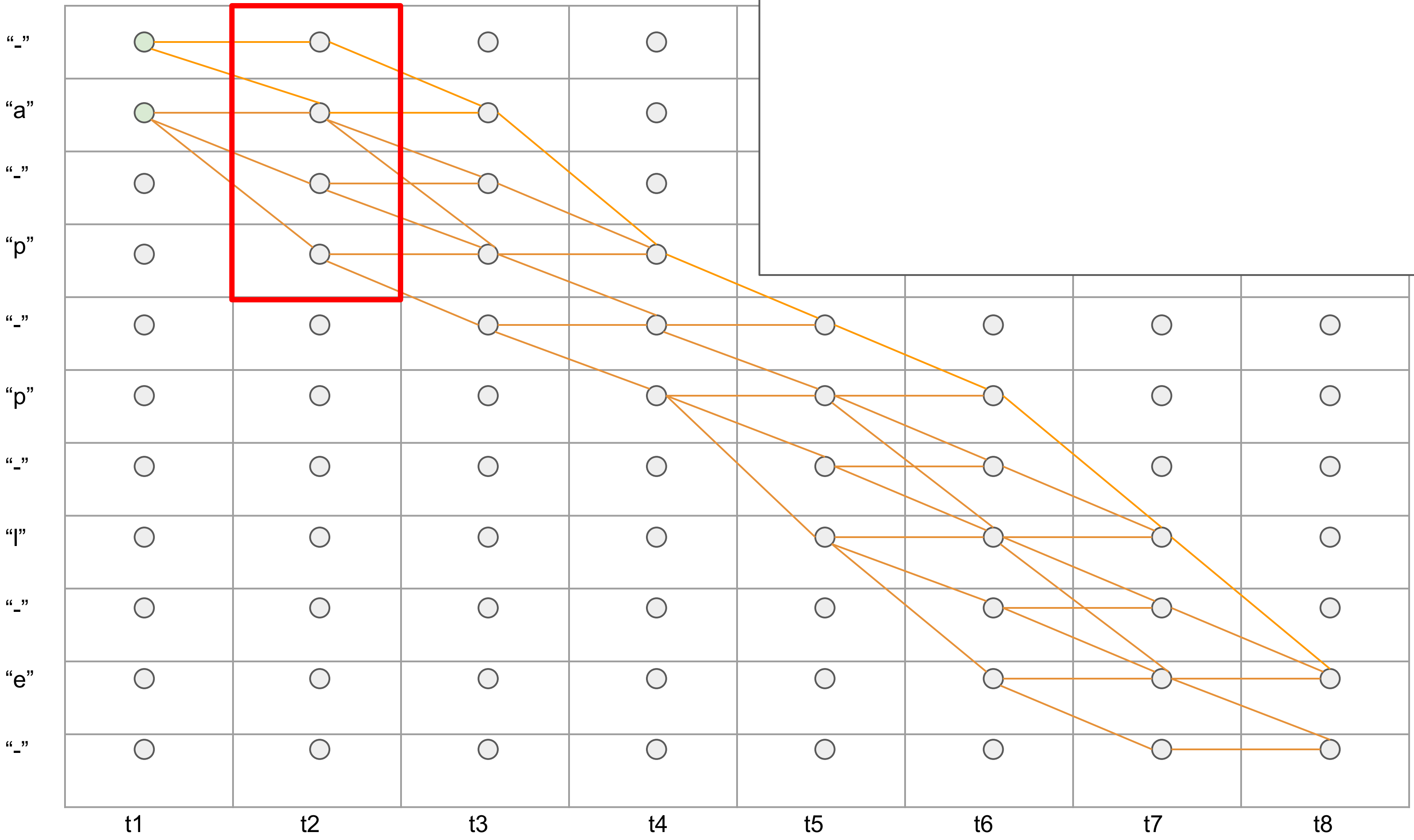
It can be applied for every time step.

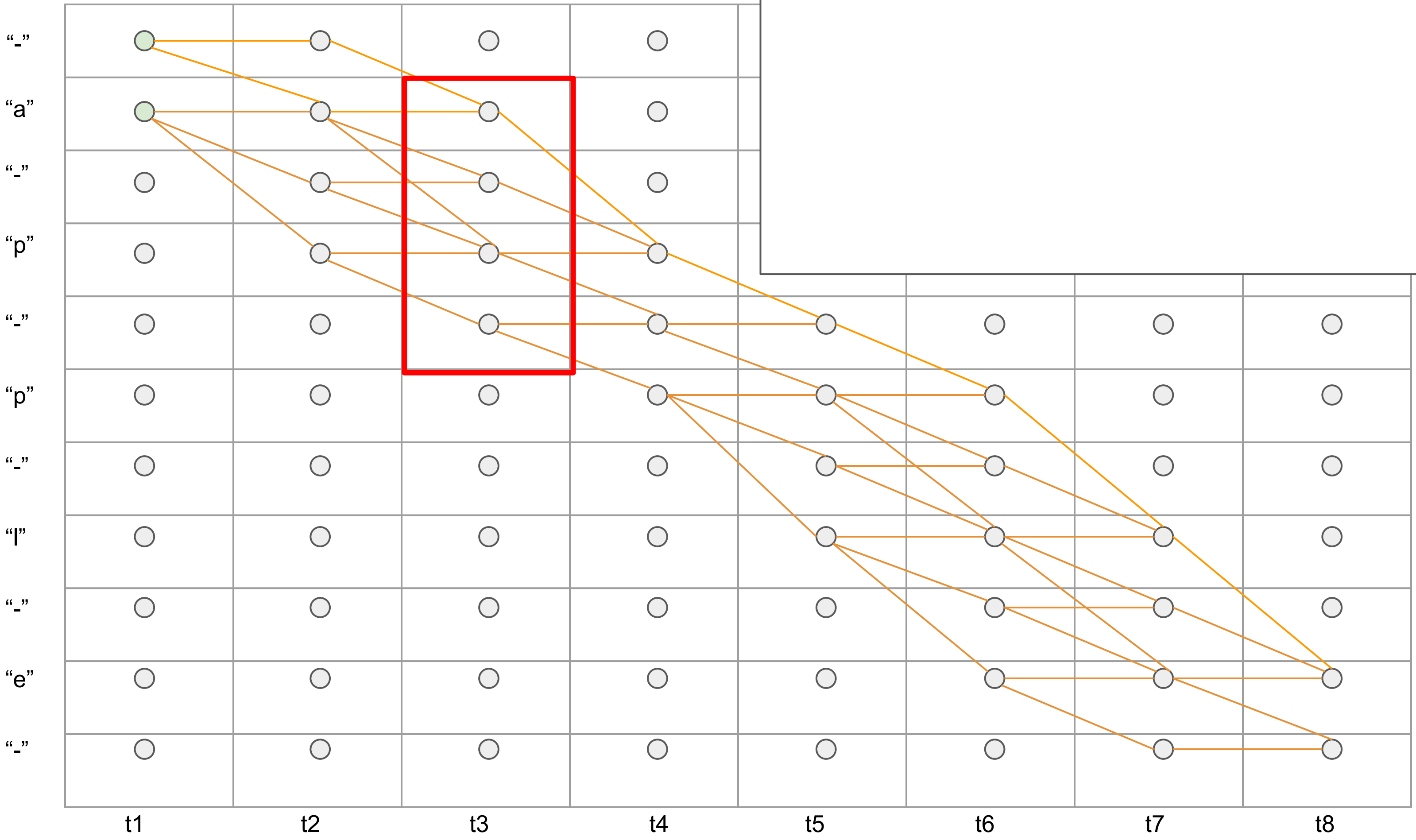




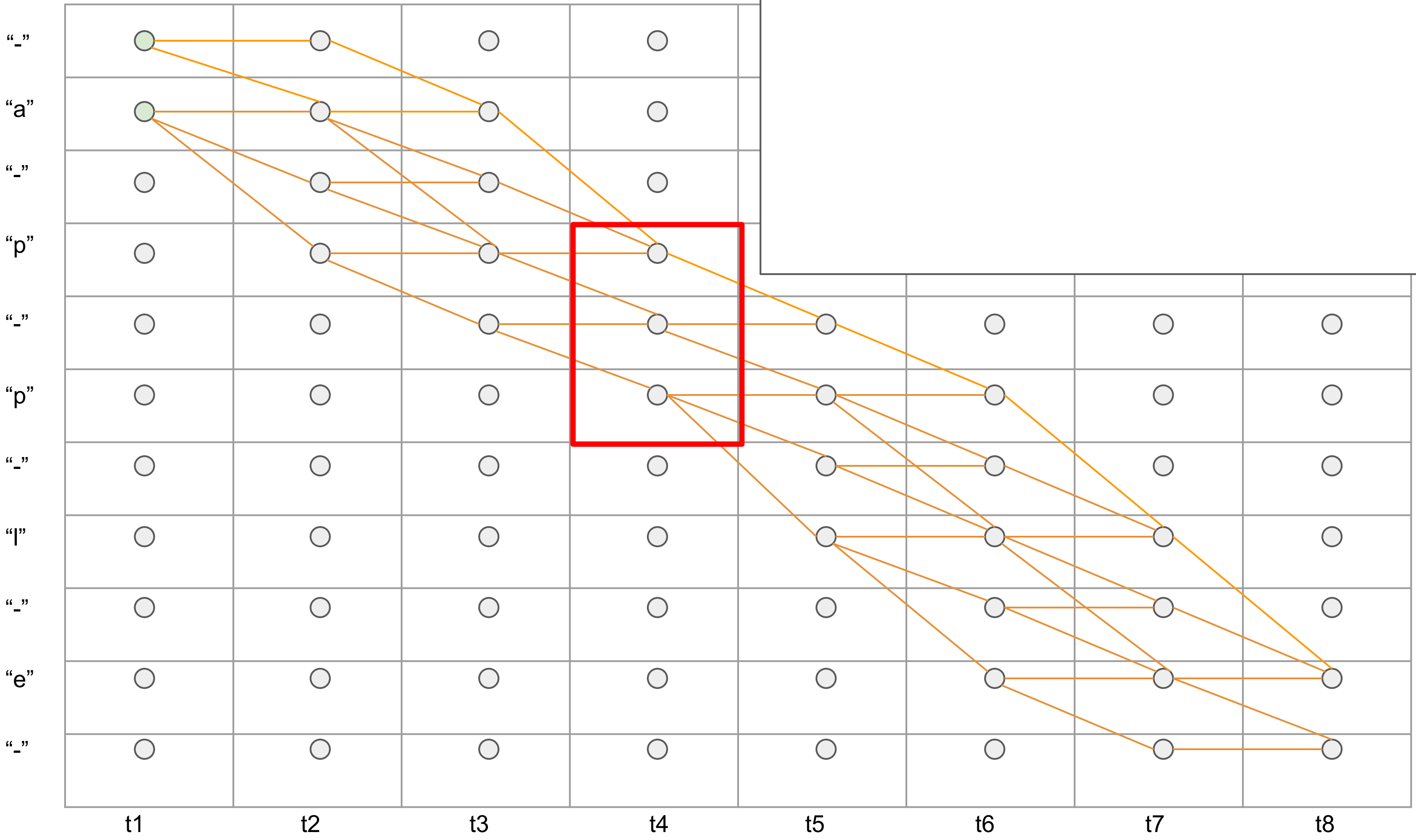
It can be applied for every time step.

It can be applied for every time step.

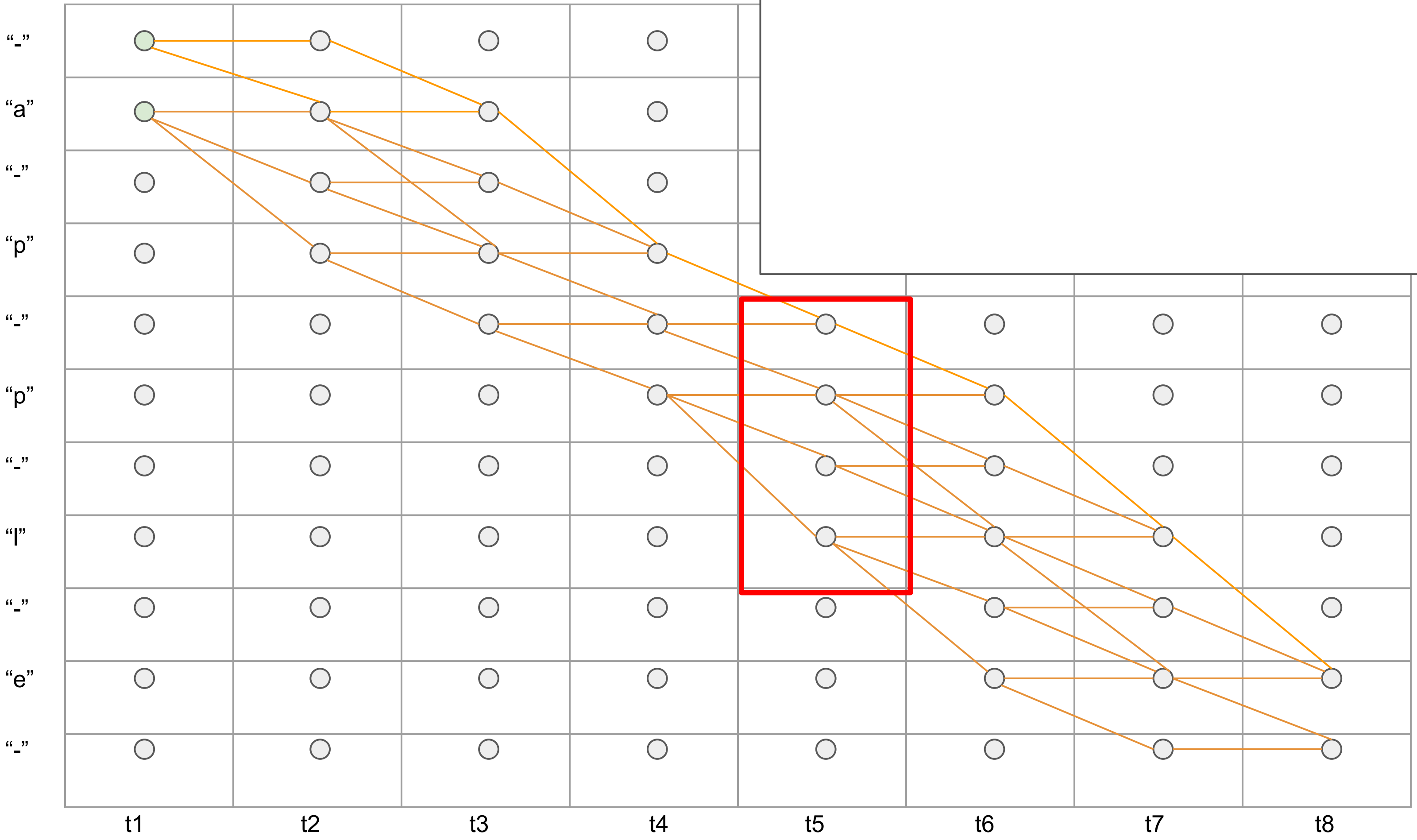




It can be applied for every time step.

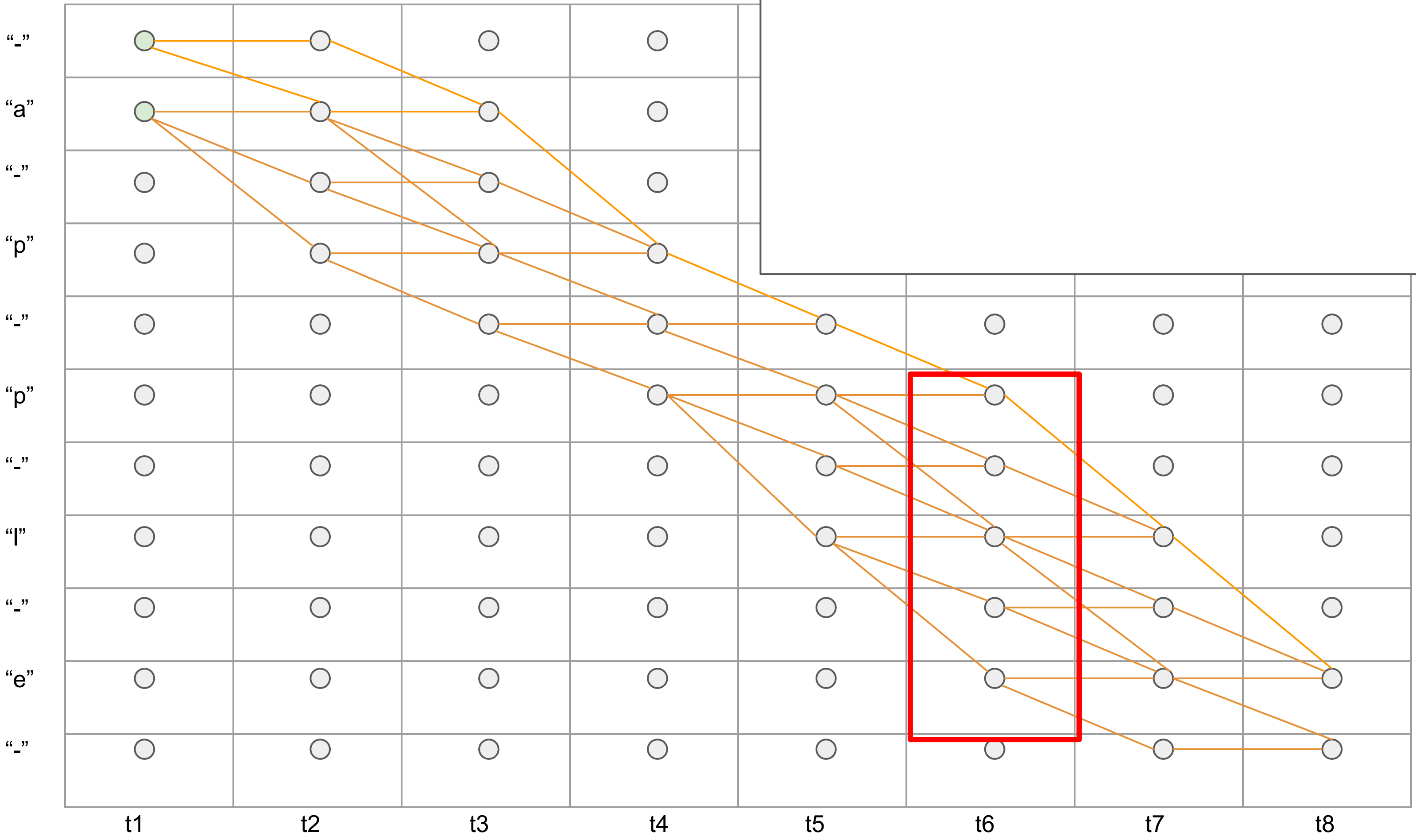


It can be applied for every time step.

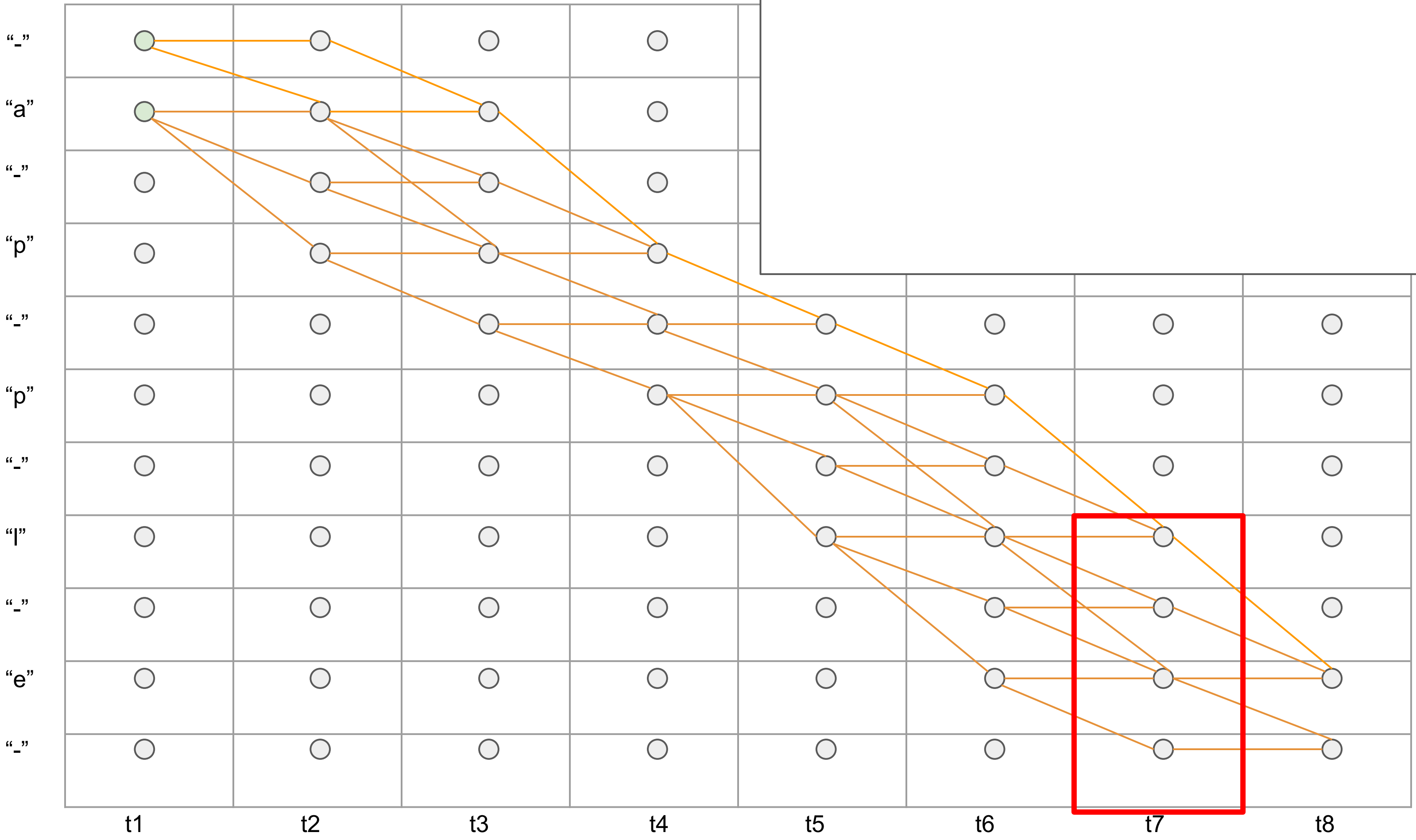


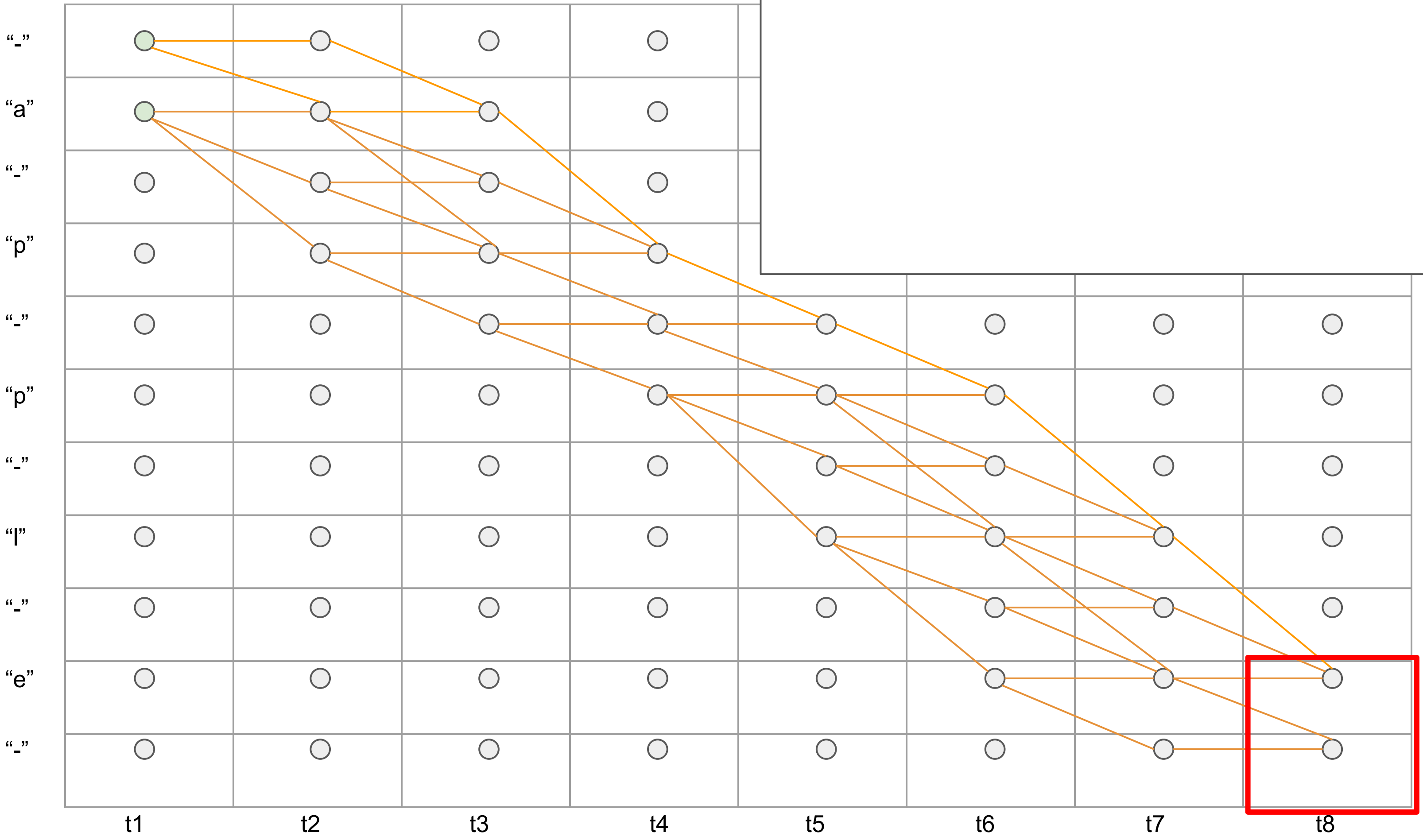
It can be applied for every time step.

It can be applied for every time step.

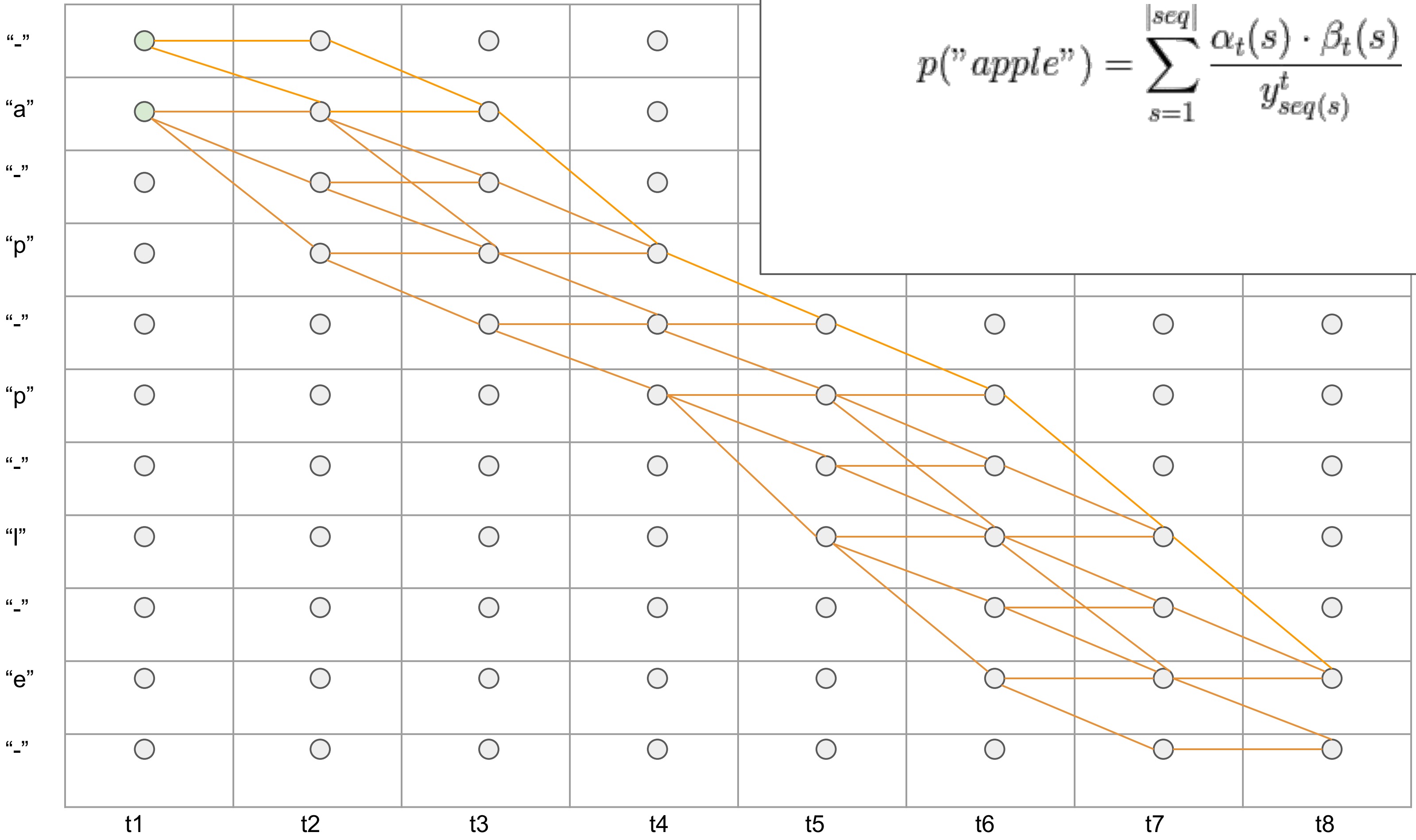


It can be applied for every time step.



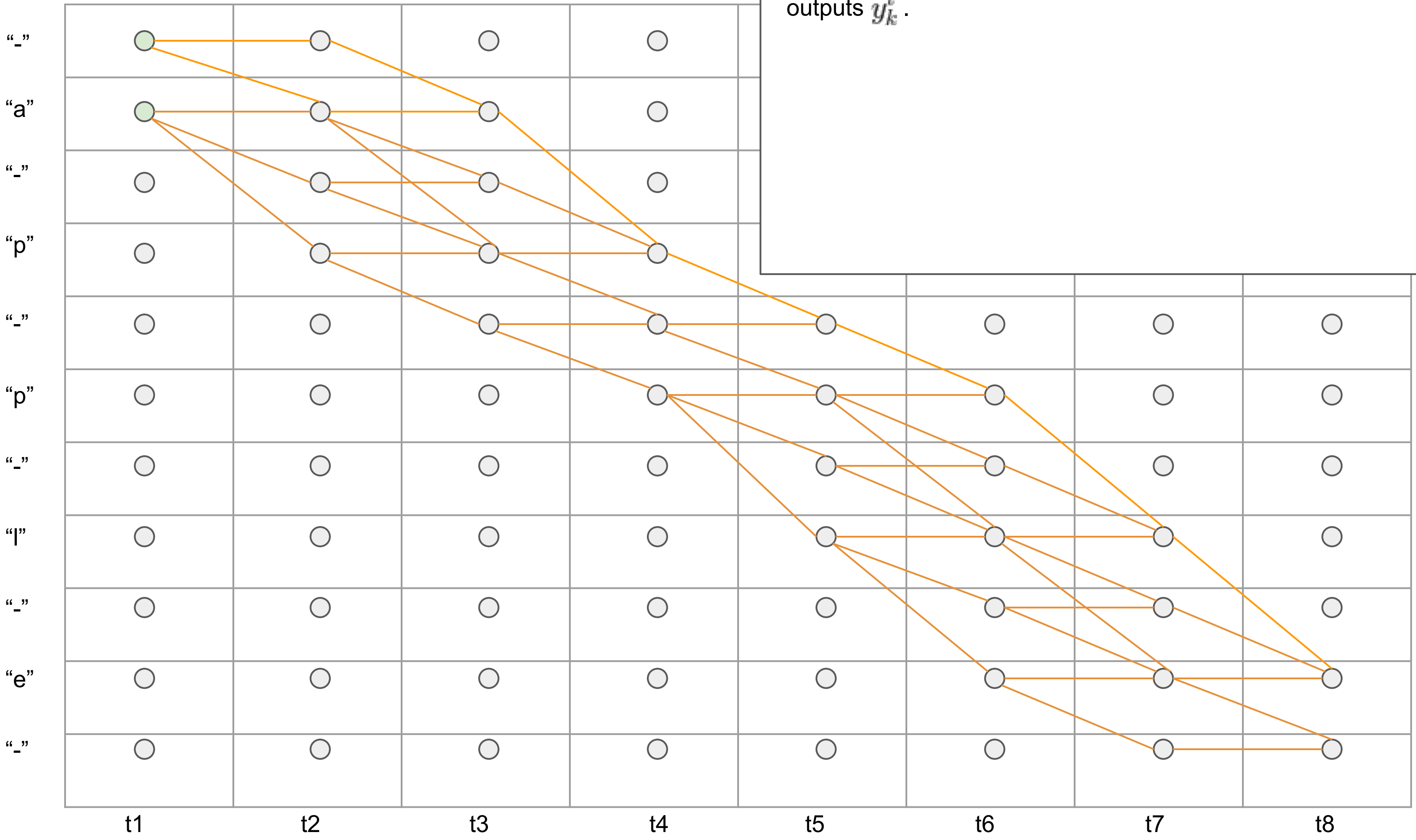


It can be applied for every time step.

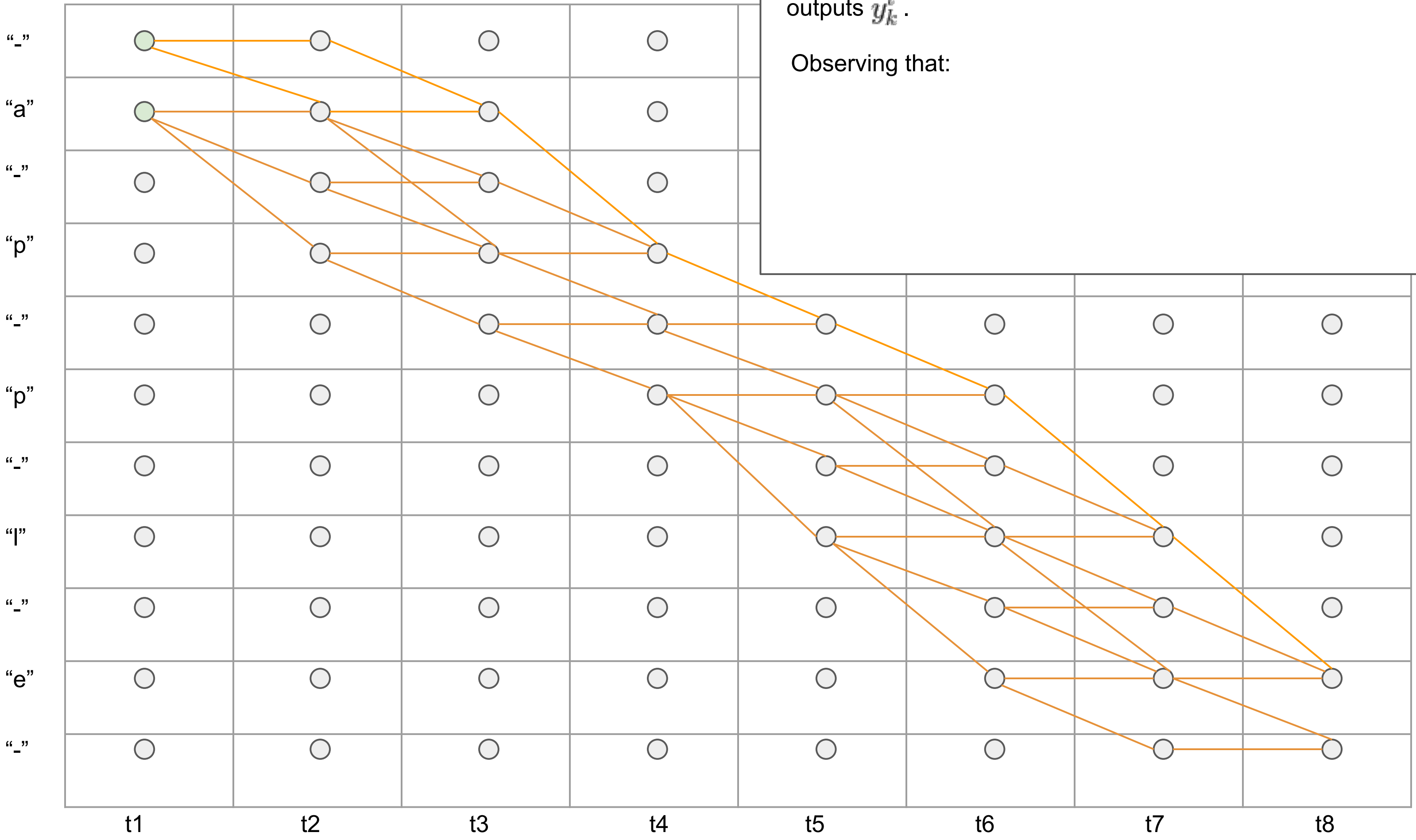


For any t, in general, probability of gt labeling will be the following:

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



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



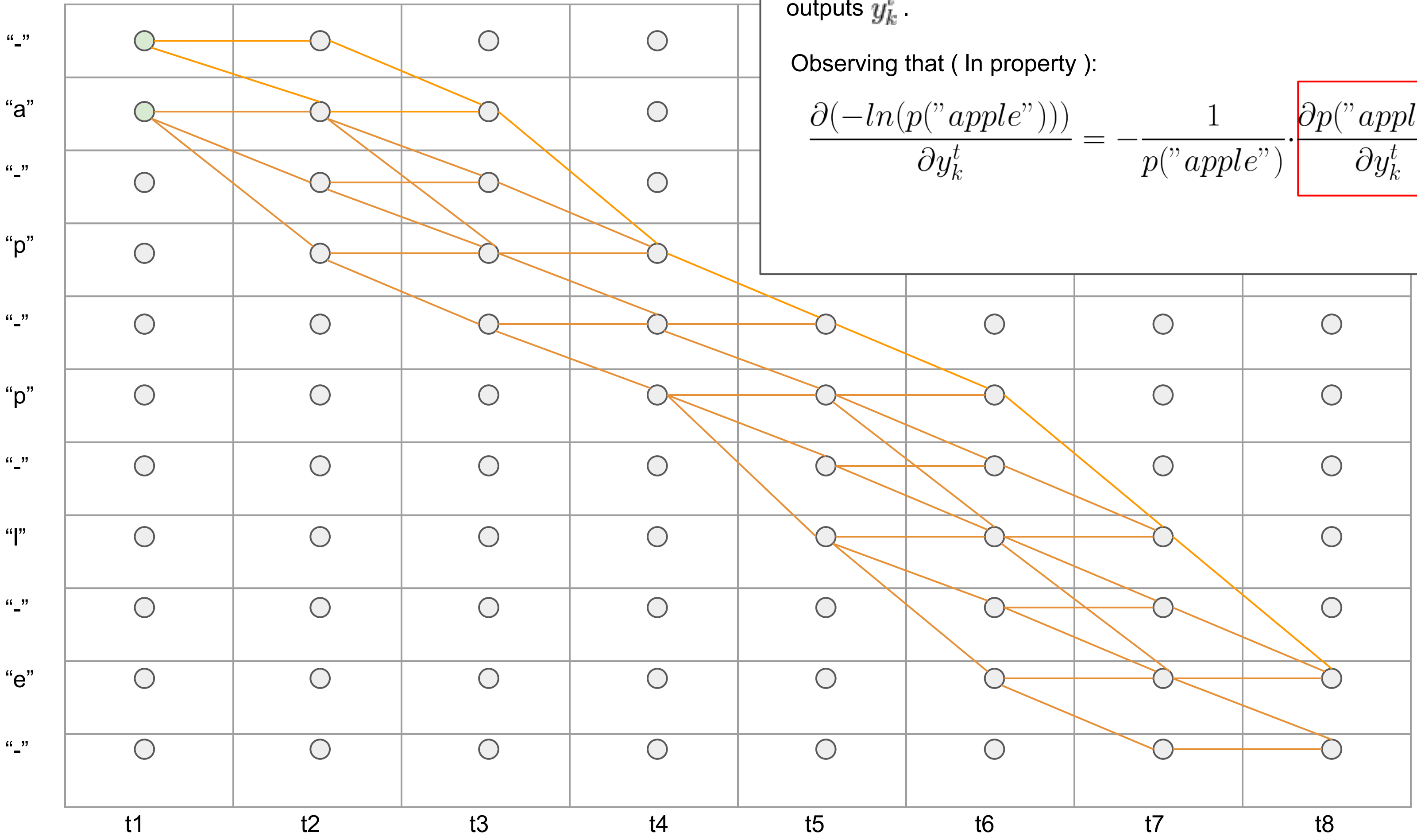
To do backprop, we should differentiate loss with respect to all network outputs $y_k^t$ .

Observing that:

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

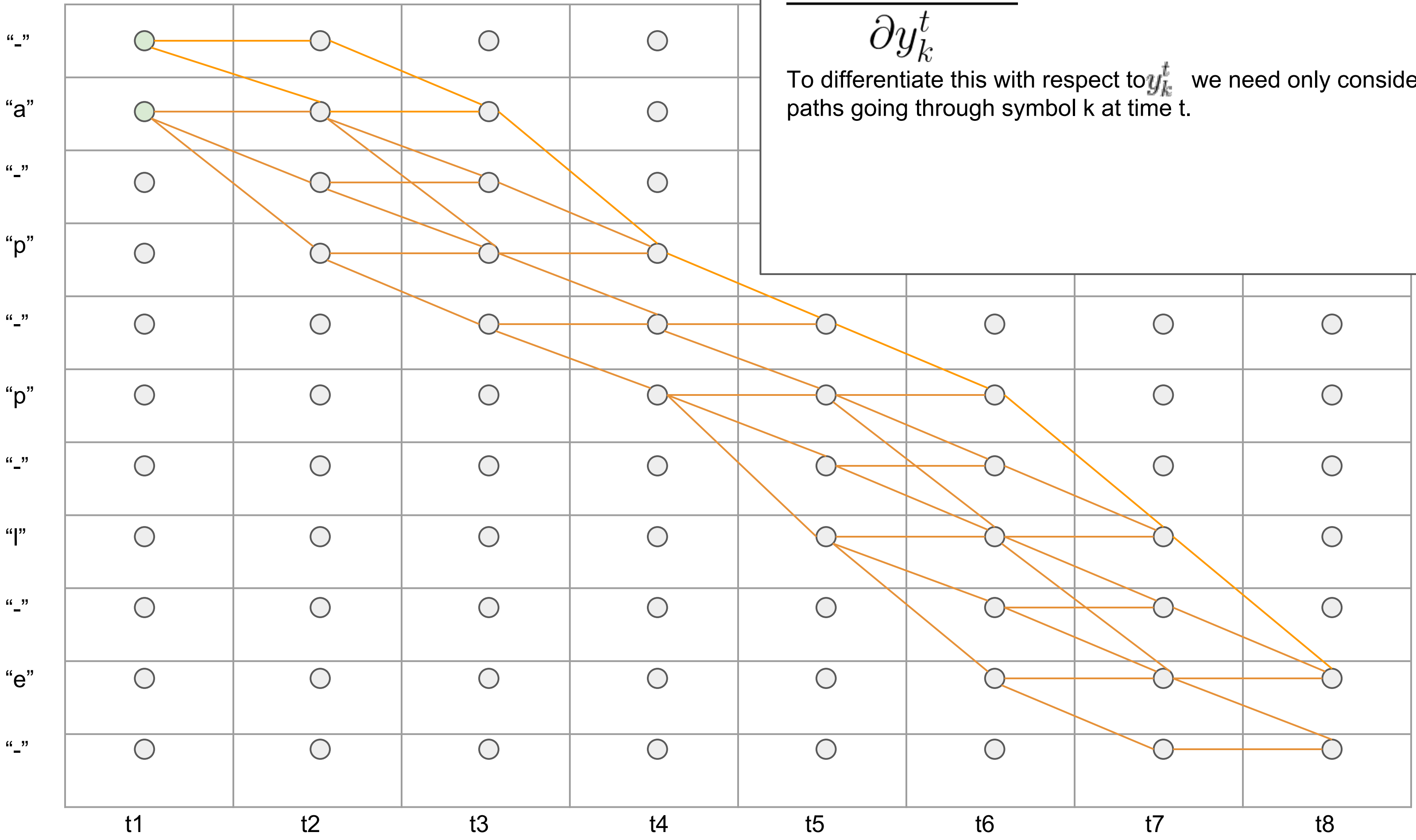
Observing that ( In property ):

$$\frac{\partial(-ln(p(''apple'')))}{\partial y_k^t} = -\frac{1}{p(''apple'')} \cdot \boxed{\frac{\partial p(''apple'')}{\partial y_k^t}}$$



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

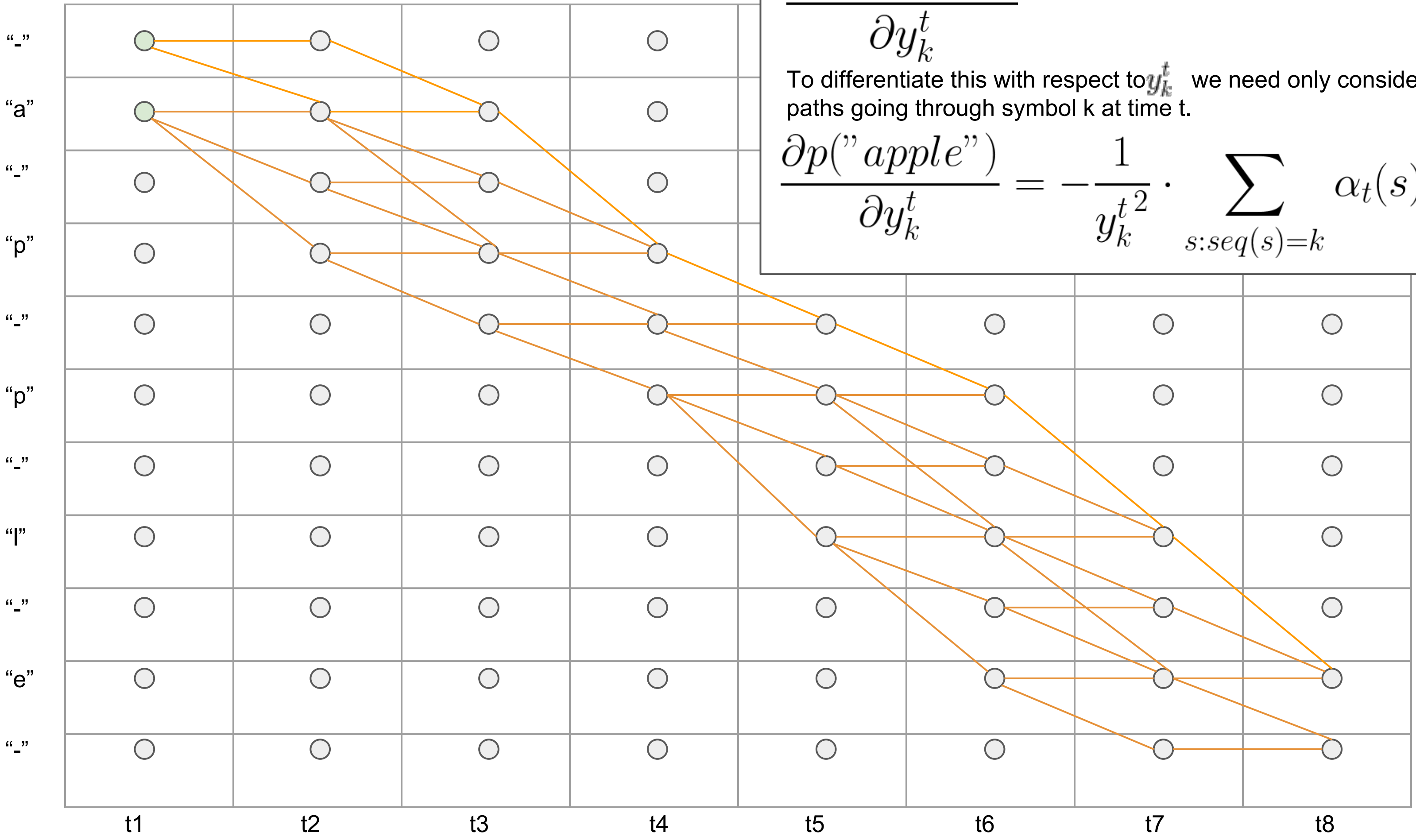
To differentiate this with respect to $y_k^t$ we need only consider those paths going through symbol k at time t.

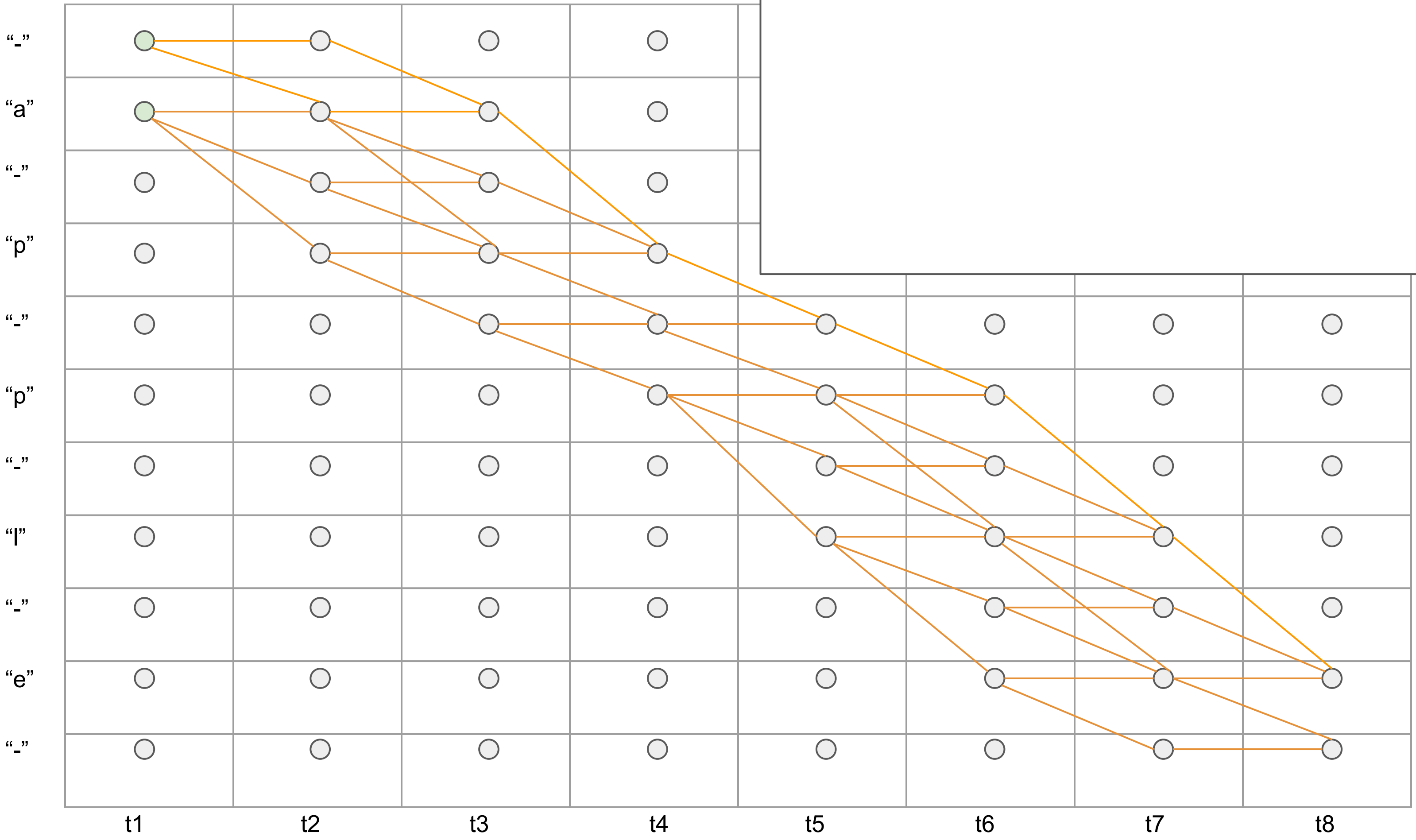


$$\frac{\partial p(\text{"}apple\text{"})}{\partial y_k^t}$$

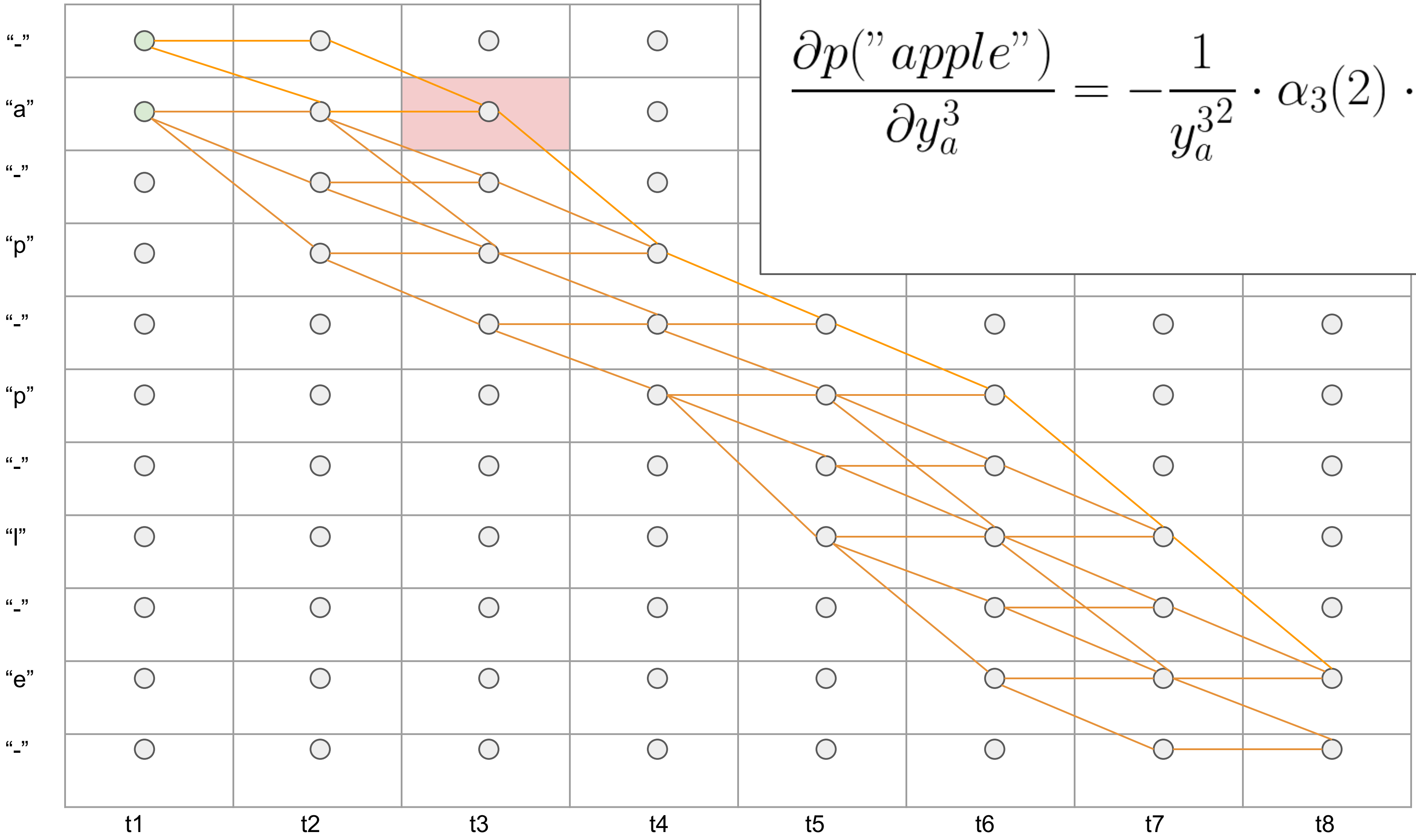
To differentiate this with respect to $y_k^t$ we need only consider those paths going through symbol k at time t.

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





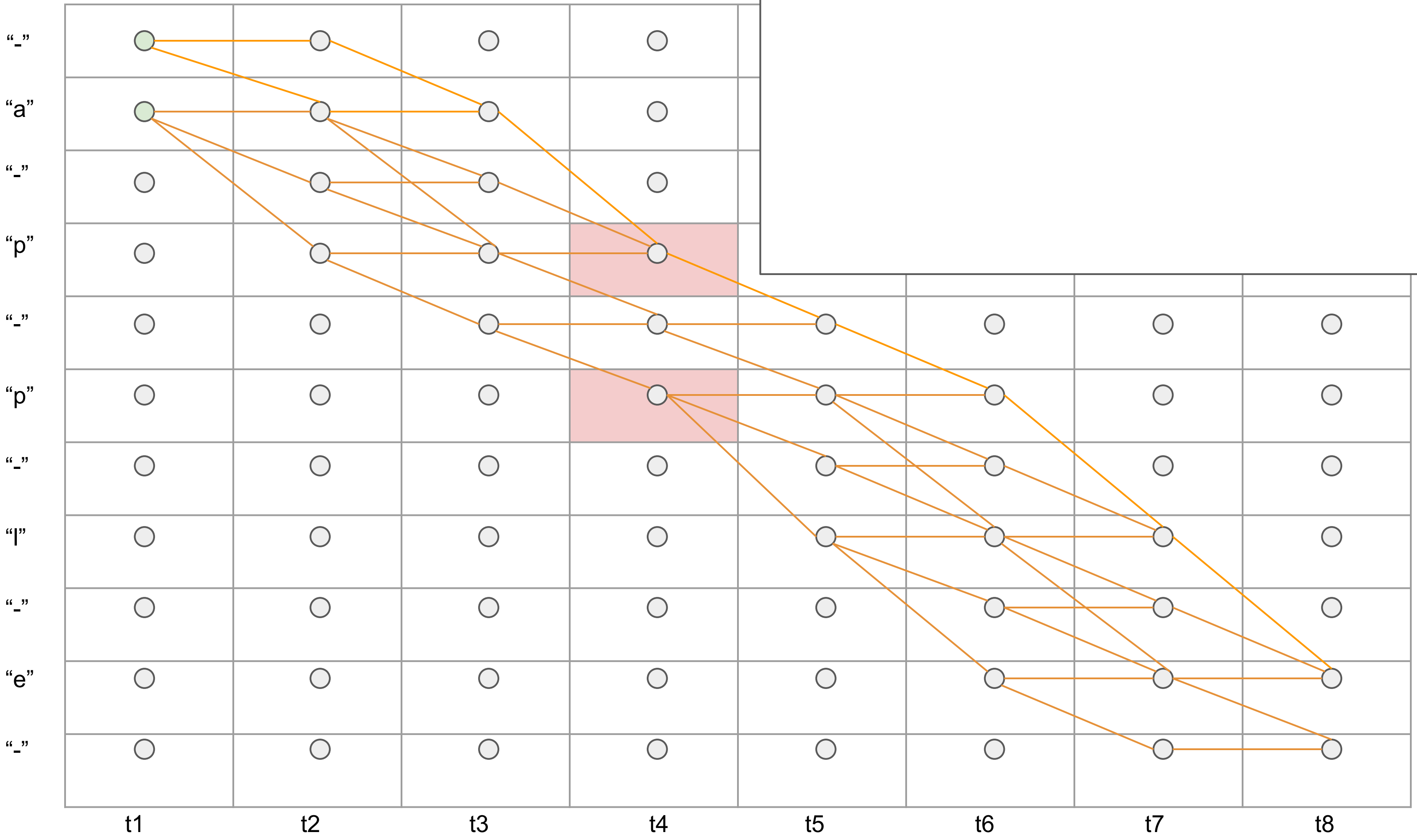
Example1:



Example1:

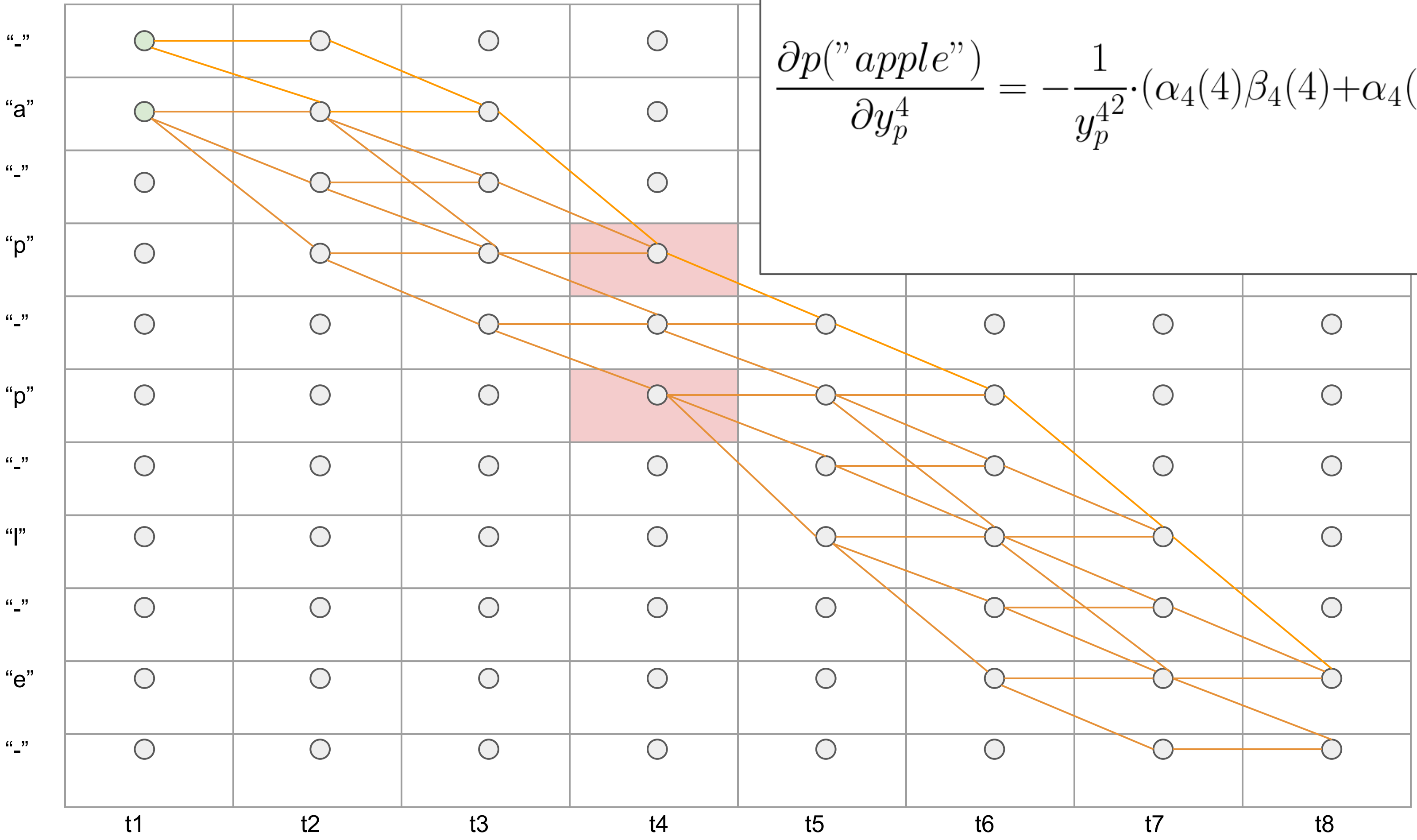
$$\frac{\partial p(\text{"}apple\text{"})}{\partial y_a^3} = -\frac{1}{{y_a^3}^2} \cdot \alpha_3(2) \cdot \beta_3(2)$$

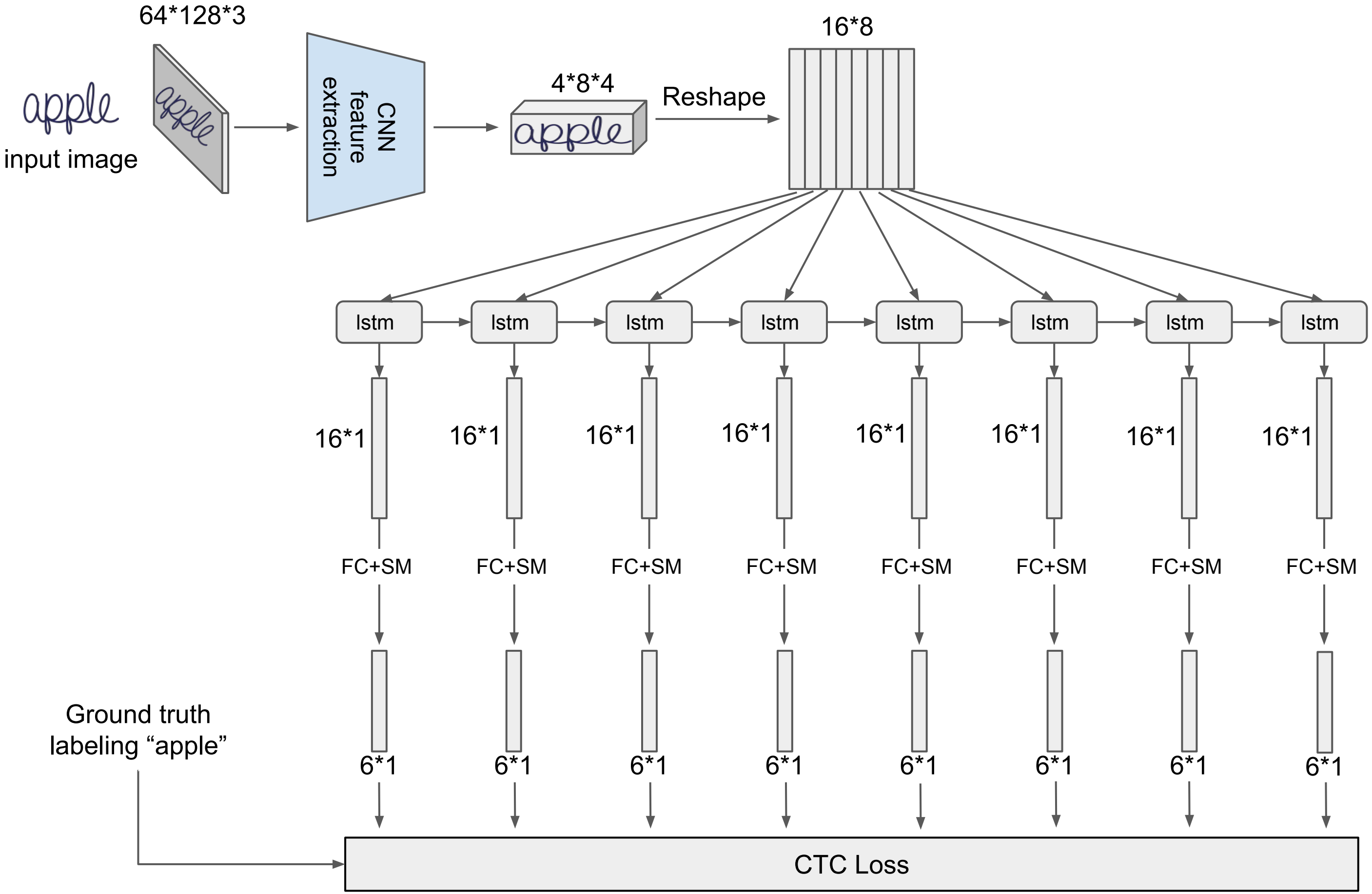
Example2:



Example2:

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




64*128*3
apple
input image
CNN
feature
extraction
4*8*4
apple
Reshape
16*8
lstm
lstm
lstm
lstm
lstm
lstm
lstm
lstm
16*1
16*1
16*1
16*1
16*1
16*1
16*1
16*1
FC+SM
FC+SM
FC+SM
FC+SM
FC+SM
FC+SM
FC+SM
FC+SM
6*1
6*1
6*1
6*1
6*1
6*1
6*1
6*1
Ground truth
labeling “apple”
CTC Loss

# Conclusion

- Dynamic programming
- Matrix $\alpha$ (forward variables) is used to compute loss
- Matrix $\beta$ (backward variables) is used to compute gradients

# Thank you

**Our Website:**

deepsystems.ai

**Products:**

supervise.ly - Dataset management, annotation and preparation service

movix.ai - Interactive, lstm-based movie recommender system

**Outsource projects:**

Our team is looking for business partners to make exciting deep learning solutions.