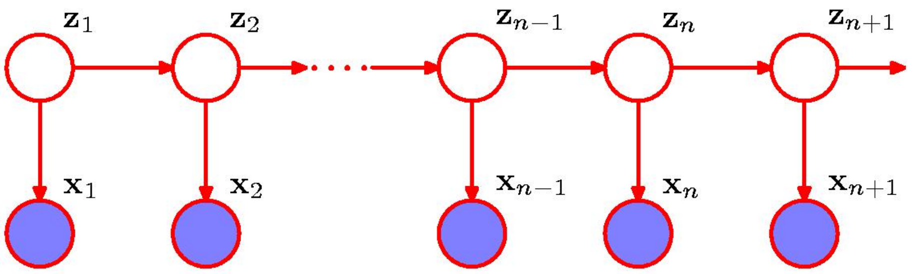
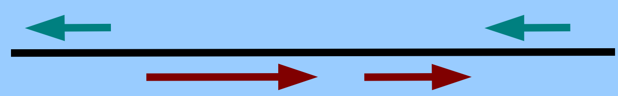
Hidden Markov Models

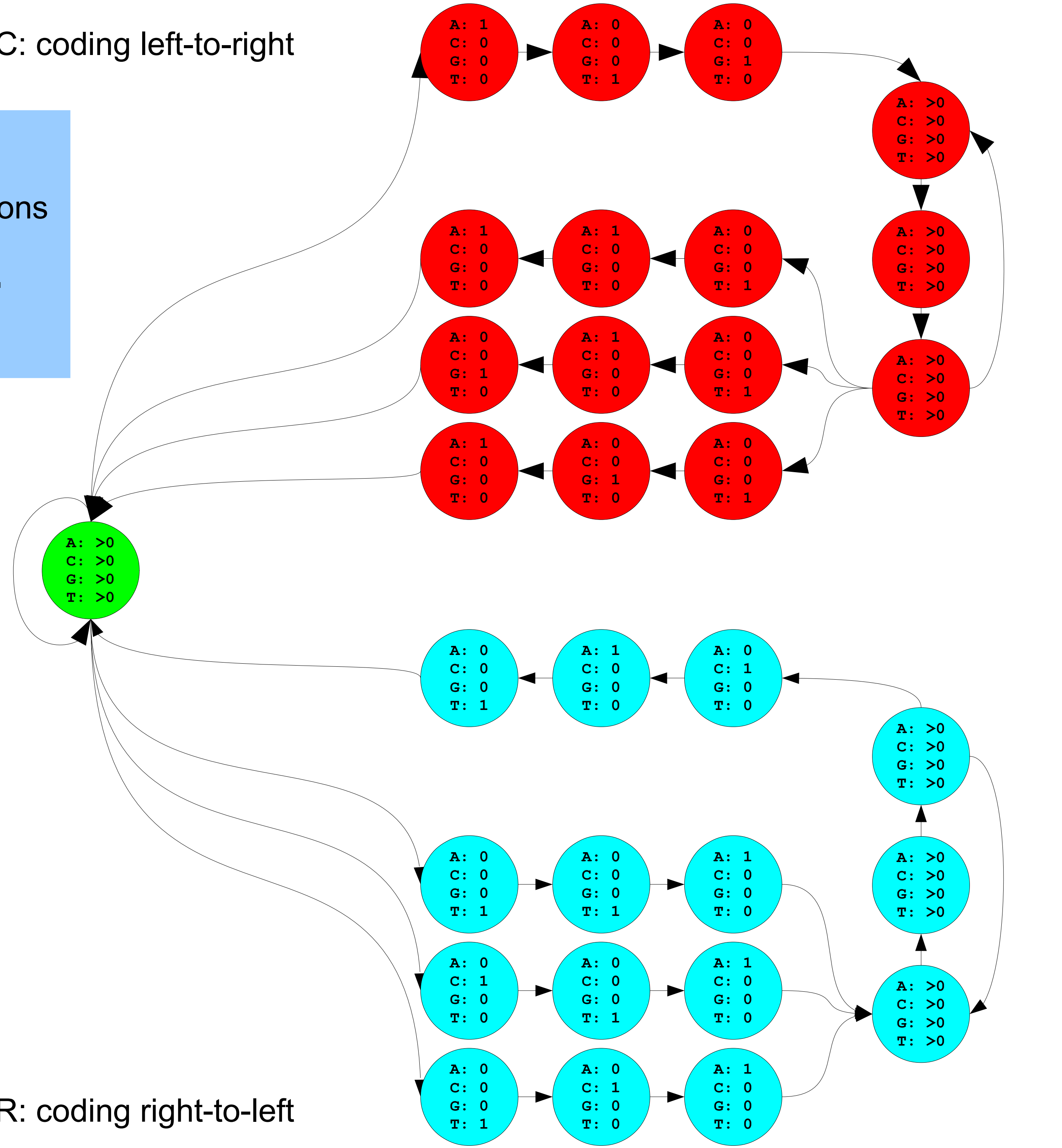
Some useful extensions



Even more biology

There can be genes in both directions





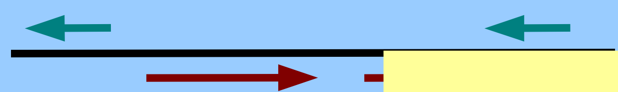
$\pi_N = 1$

$\pi_C = 0$

C: coding left-to-right

Even more biology

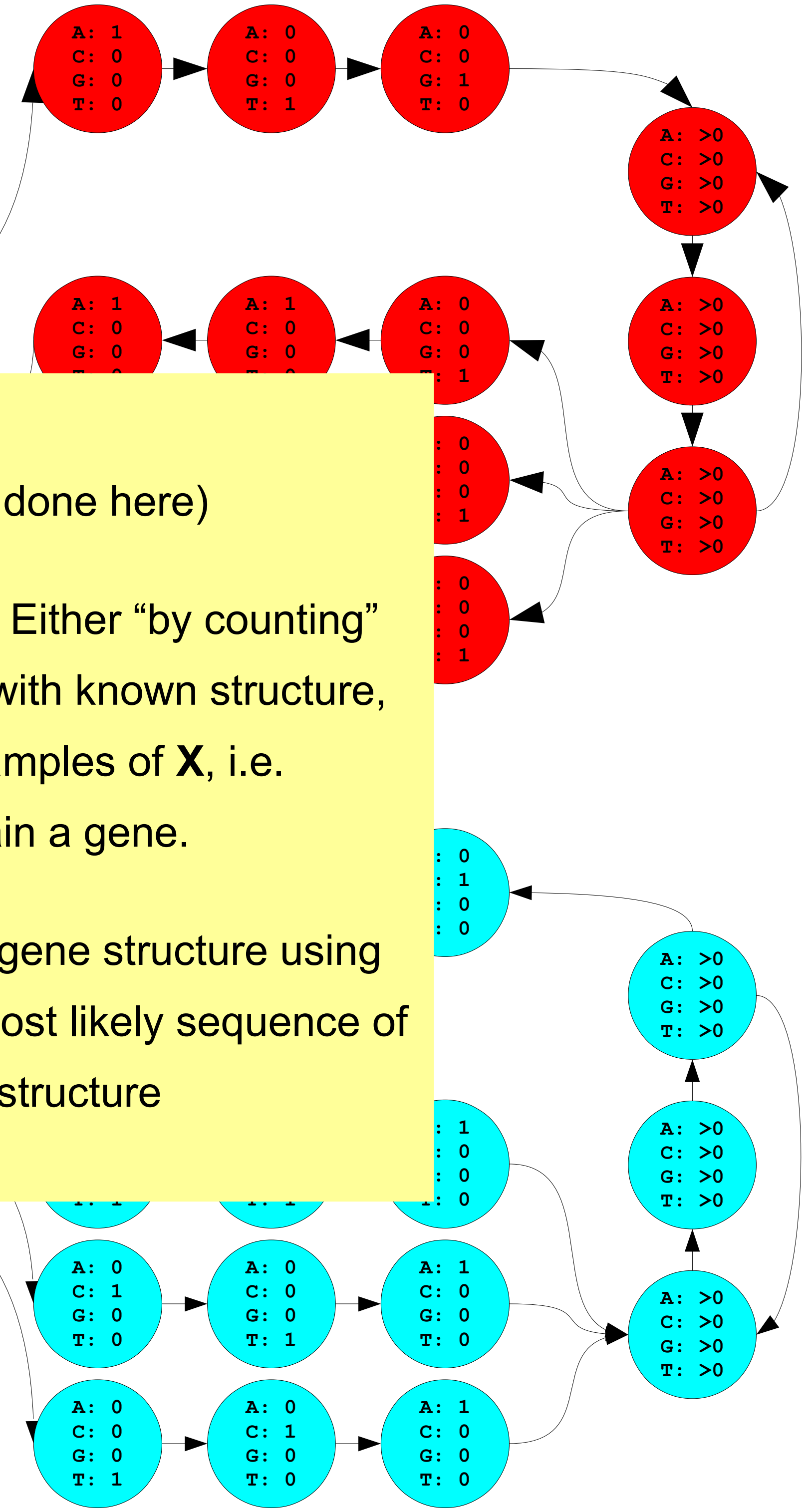
There can be genes in both directions



Gene finding

- Select initial model structure (e.g. as done here)
- Select model parameters by training. Either "by counting" from examples of (**X**,**Z**)'s, i.e. genes with known structure, or by EM- or Viterbi-training from examples of **X**, i.e. sequences which are known to contain a gene.
- Given a new sequence **X**, predict its gene structure using the Viterbi algorithm for finding the most likely sequence of underlying latent states, i.e. its gene structure

N: No

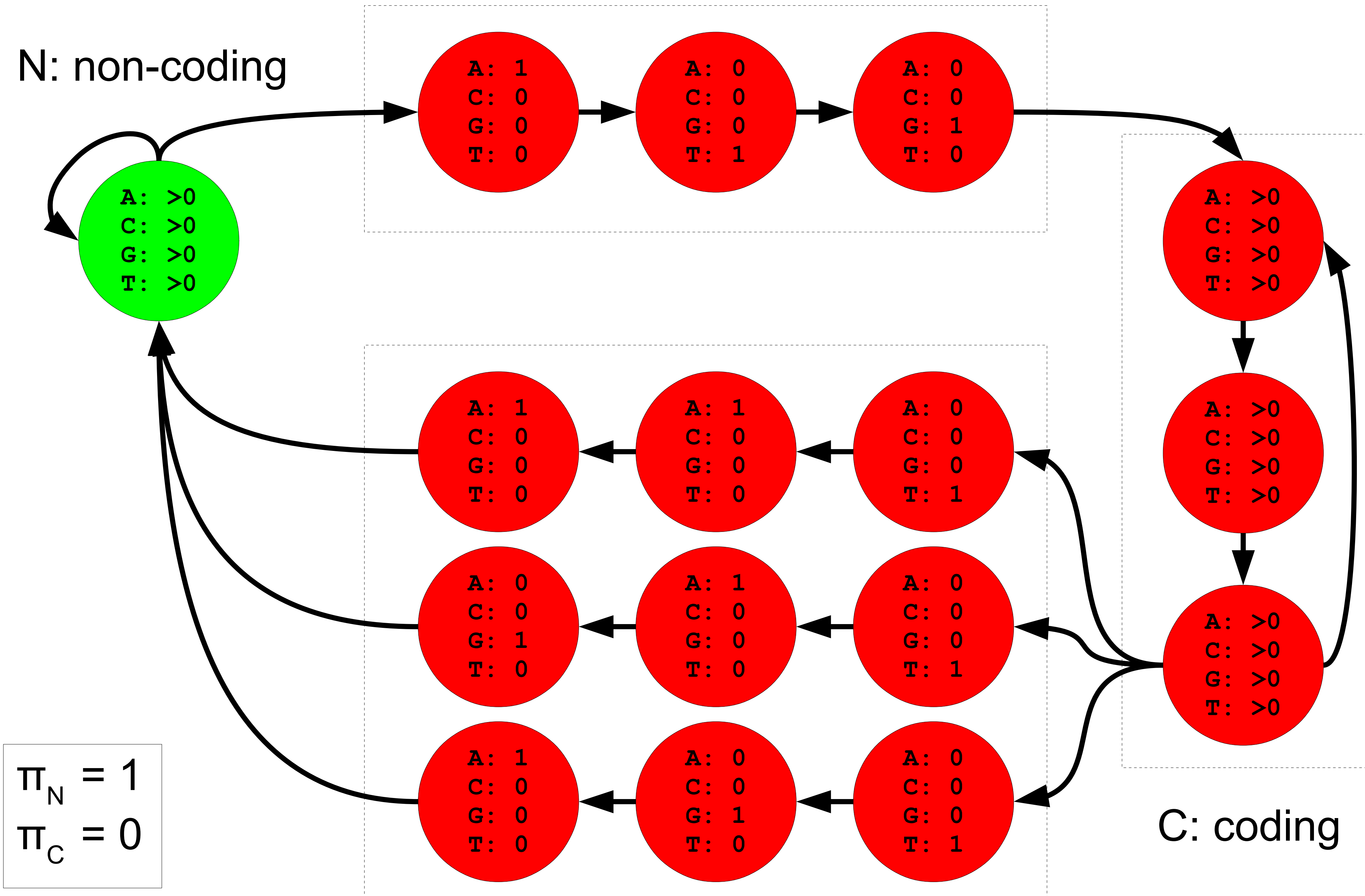


R: coding right-to-left

$\pi_N = 1$

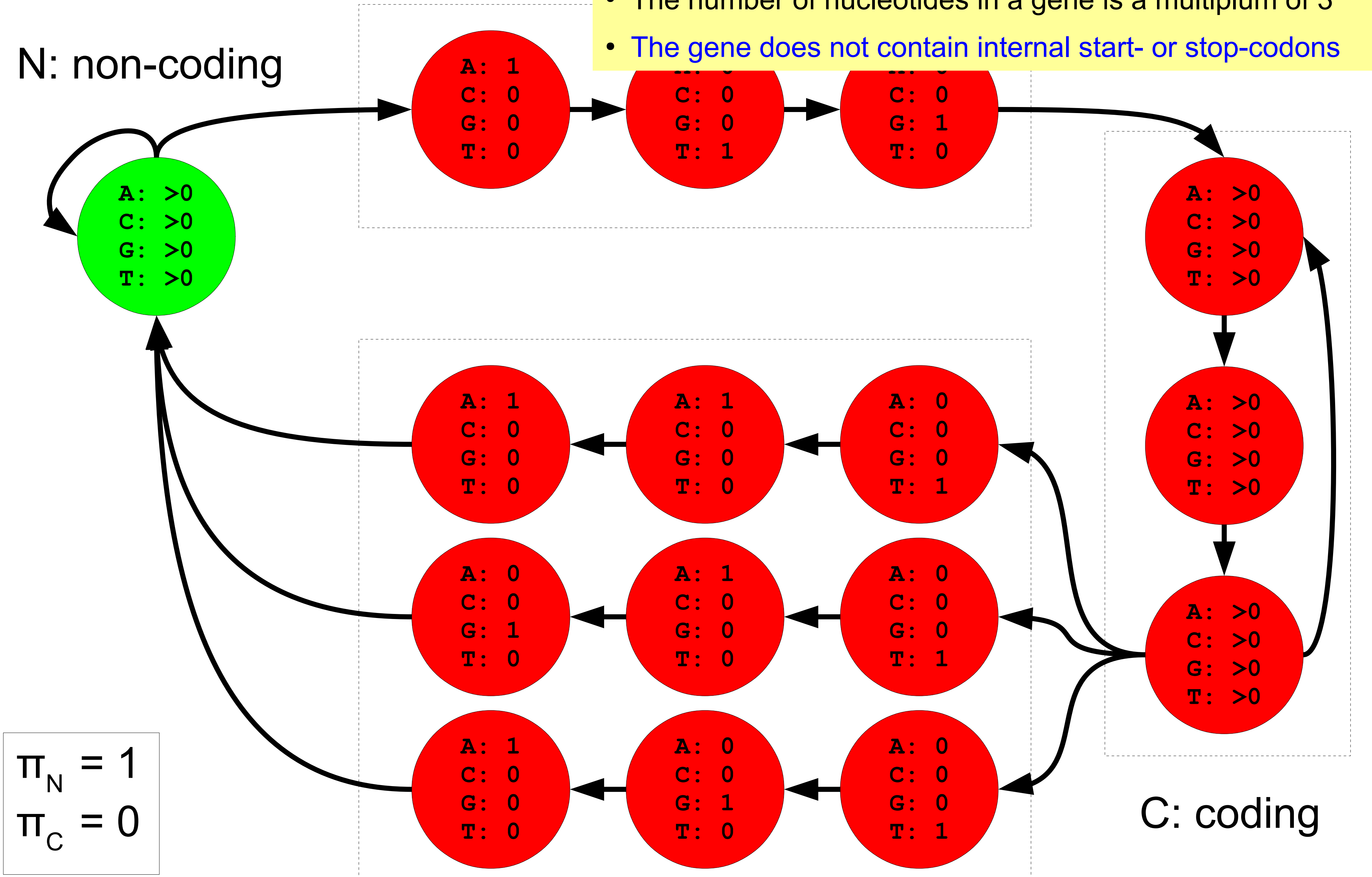
$\pi_C = 0$

The "forward-coding" part



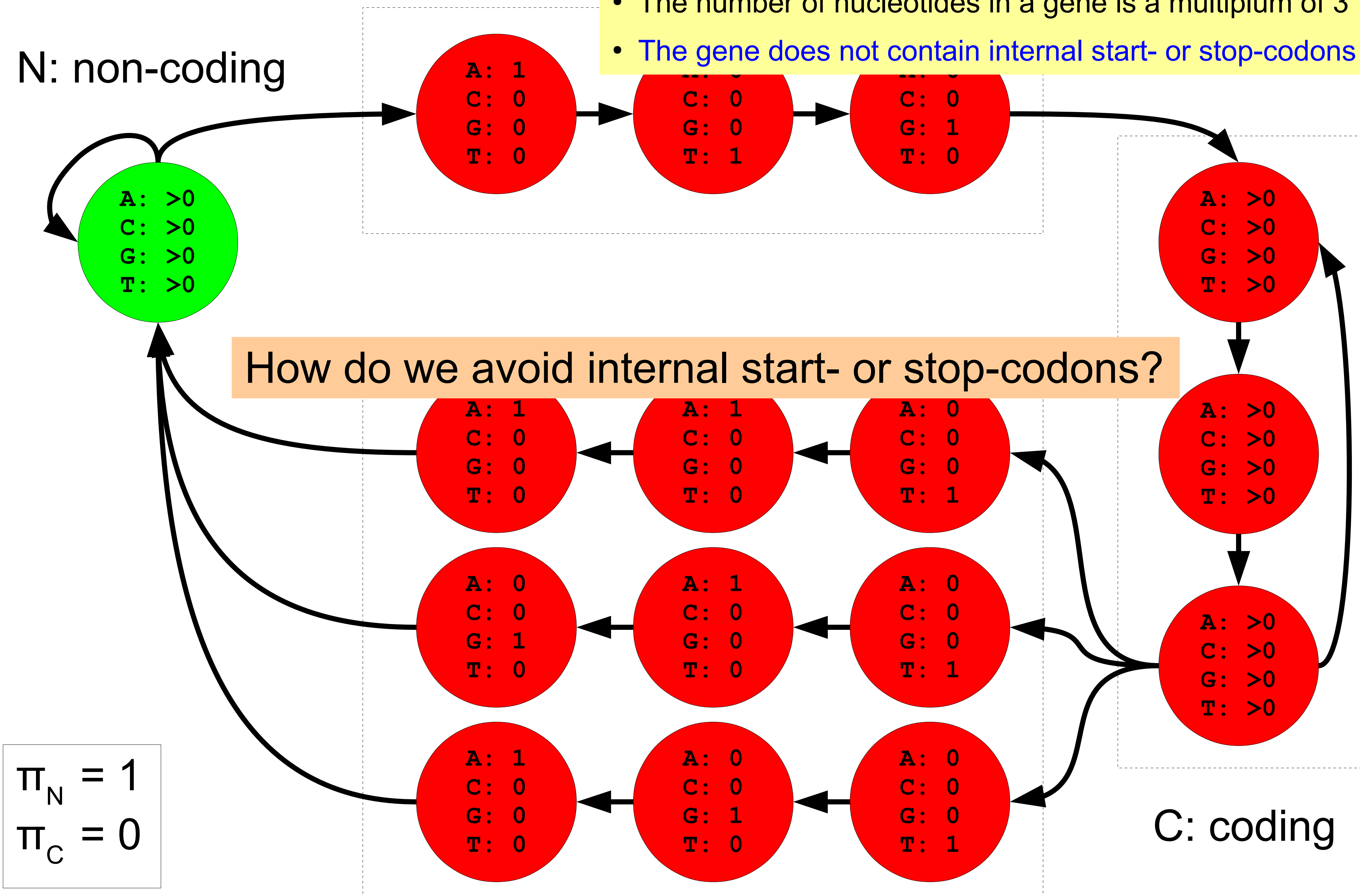
The “forwa

- The gene is a substring of the DNA sequence of A,C,G,T's
- The gene starts with a start-codon **atg**
- The gene ends with a stop-codon **taa**, **tag** or **tga**
- The number of nucleotides in a gene is a multiplum of 3
- The gene does not contain internal start- or stop-codons



The "forwa

- The gene is a substring of the DNA sequence of A,C,G,T's
- The gene starts with a start-codon **atg**
- The gene ends with a stop-codon **taa**, **tag** or **tga**
- The number of nucleotides in a gene is a multiplum of 3
- The gene does not contain internal start- or stop-codons

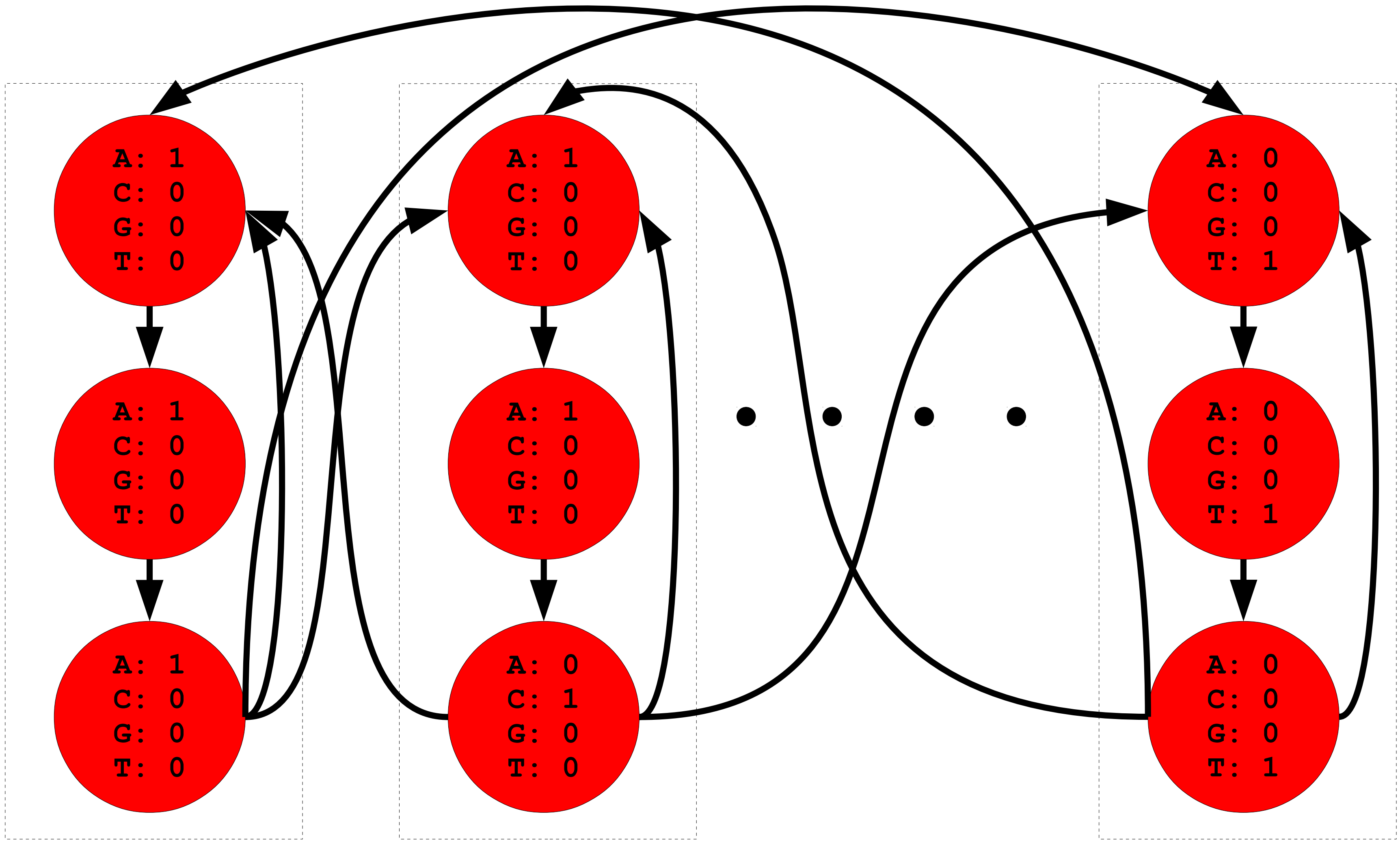


How do we avoid internal start- or stop-codons?

$\pi_N = 1$

$\pi_C = 0$

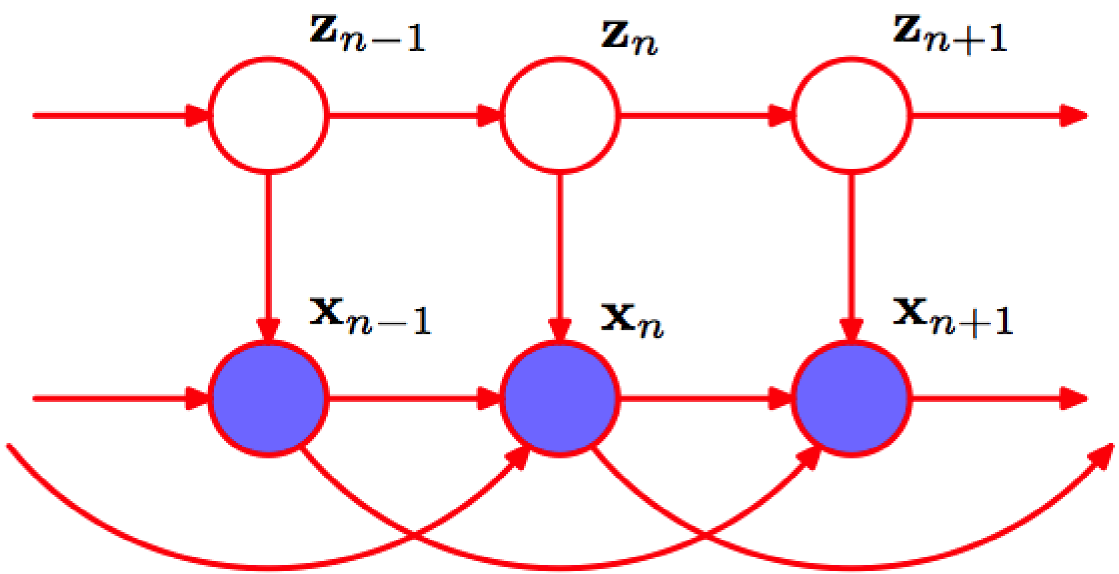
Avoiding internal start- or stop-codons



Encode the emission of each legal codon as a sequence of states.
Many states (60*3=180) and non-trivial transitions (60*59=3540)!

Other ideas?

Autoregressive HMMs



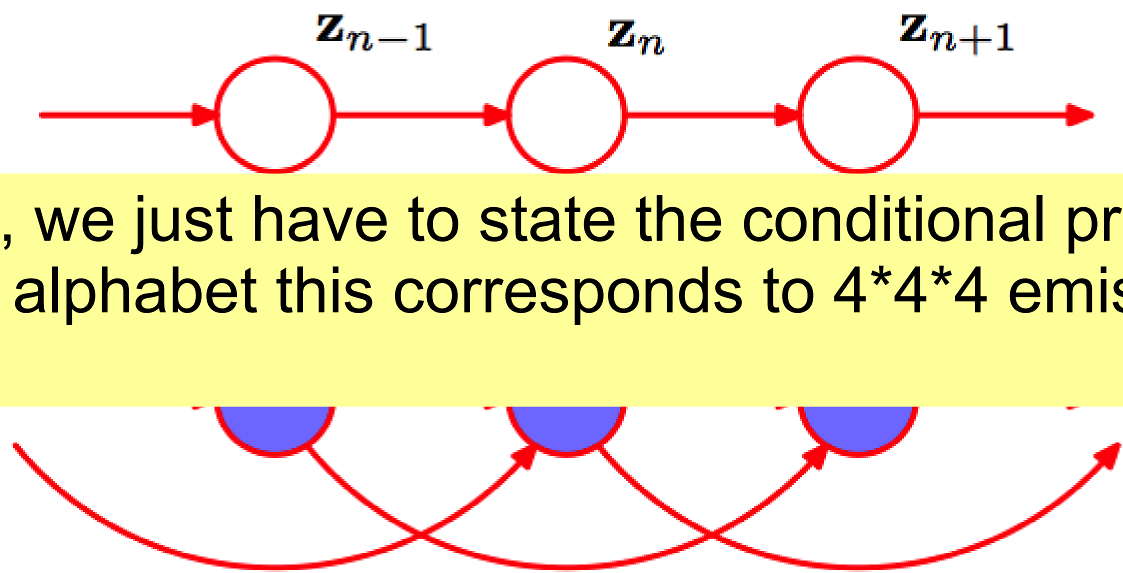
The probability of emitting $\mathbf{x}_n$ depends also on $\mathbf{x}_{n\text{-}1}$ and $\mathbf{x}_{n\text{-}2}$

The basic algorithms remain the same:

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

$$\omega(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \mathbf{z}_n) \max_{\mathbf{z}_{n-1}} \omega(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

Autoregressive HMMs



For each state, we just have to state the conditional probabilities. For a 4-letter DNA alphabet this corresponds to 4*4*4 emission prob.

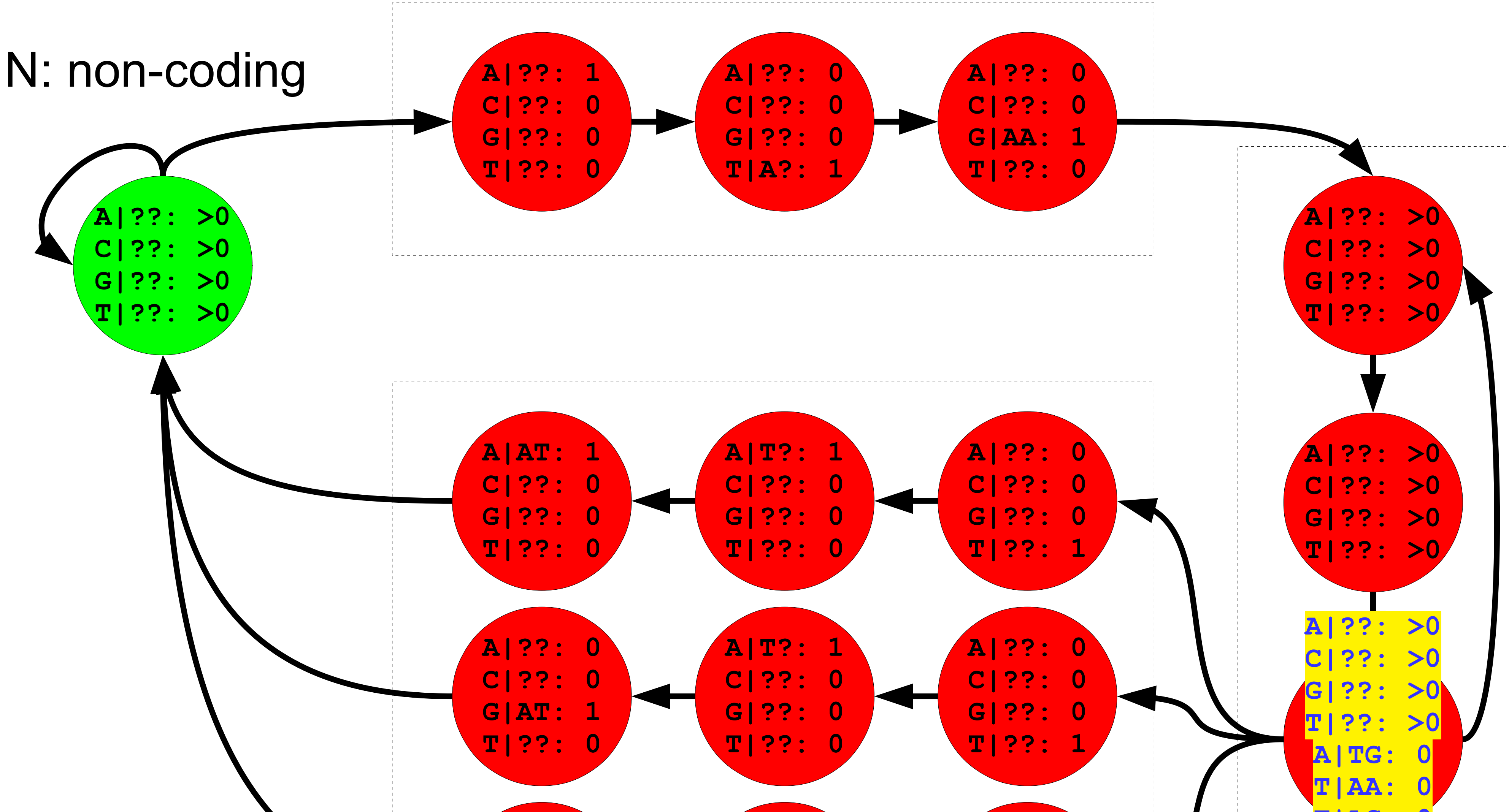
The probability of emitting $\mathbf{x}_n$ depends also on $\mathbf{x}_{n-1}$ and $\mathbf{x}_{n-2}$

The basic algorithms remain the same:

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

$$\omega(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \mathbf{z}_n) \max_{\mathbf{z}_{n-1}} \omega(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

Adjusting our simple HMM



$\pi_N = 1$

$\pi_C = 0$

Emitting a variable number of symbols

Make it possible to emit a variable number of symbols depending on the state. Fx when being in state $\mathbf{z}_n$ the model emits $d_{\mathbf{z}n}$ symbols, where $d_{\mathbf{z}n}$ is an integer ≥ 0.

The basic algorithms can easily be reformulated, fx Viterbi:

Emitting a variable number of symbols

Make it possible to emit a variable number of symbols depending on the state. Fx when being in state $\mathbf{z}_n$ the model emits $d_{\mathbf{z}n}$ symbols, where $d_{\mathbf{z}n}$ is an integer ≥ 0.

The basic algorithms can easily be reformulated, fx Viterbi:

$\omega(n, k)$: The probability of the most likely path generating the first *n* symbols and ending in state *k*.

$$\omega(n, k) = \max_{k' \to k} \omega(n - d_k, k') p(k' \to k) p(\mathbf{x}_n \ldots \mathbf{x}_{n-d_k+1} | k)$$

Emitting a variable number of symbols

Make it possible to emit a variable number of symbols depending on the state. Fx when being in state $\mathbf{z}_n$ the model emits $d_{\mathbf{z}n}$ symbols, where $d_{\mathbf{z}n}$ is an integer ≥ 0.

The basic algorithms can easily be reformulated, fx Viterbi:

$\omega(n, k)$: The probability of the most likely path generating the first *n* symbols and ending in state *k*.

$$\omega(n, k) = \max_{k' \to k} \omega(n - d_k, k') p(k' \to k) p(\mathbf{x}_n \ldots \mathbf{x}_{n-d_k+1} | k)$$

Transition prob from state *k'* to *k*

Emission prob of emitting d_k symbols from state *k*.

Emitting a variable number of symbols

Make it possible to emit a variable number of symbols depending on the state. Fx when being in state $\mathbf{z}_n$ the model emits $d_{\mathbf{z}n}$ symbols, where $d_{\mathbf{z}n}$ is an integer ≥ 0.

The basic algorithms can easily be reformulated, fx Viterbi:

$\omega(n,k)$: The probability of the mos[t] ... the first *n* symbols and en[d] ...

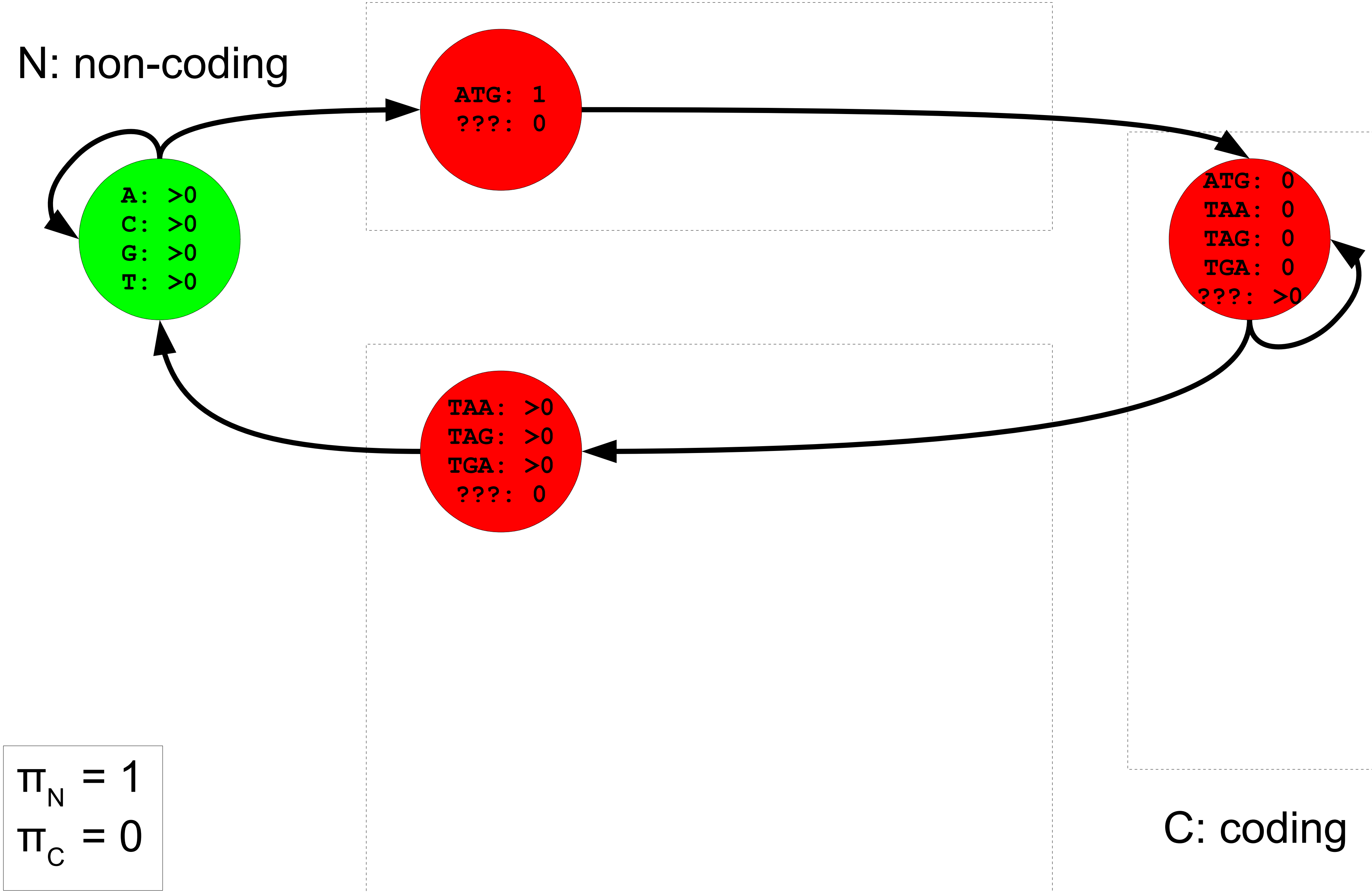
Special case: If d_k = 0 then the state is called a *silent state*.

$$\omega(n,k) = \max_{k' \to k} \omega(n-d_k, k') p(k' \to k) p(\mathbf{x}_n \ldots \mathbf{x}_{n-d_k+1} | k)$$

Transition prob from state *k'* to *k*

Emission prob of emitting d_k symbols from state *k*.

Adjusting our simple HMM



History and applications of HMMs

History of HMMs

Hidden Markov Models were introduced in statistical papers by Leonard E. Baum and others in the late1960s. One of the first applications of HMMs was speech recognition in the mid-1970s.

In the late 1980s, HMMs were applied to the analysis of biological sequences. Since then, many applications in bioinformatics...

Applications of HMMs in bioinformatics

prediction of protein-coding regions in genome sequences

modeling families of related DNA or protein sequences

prediction of secondary structure elements in proteins

... and many others ...