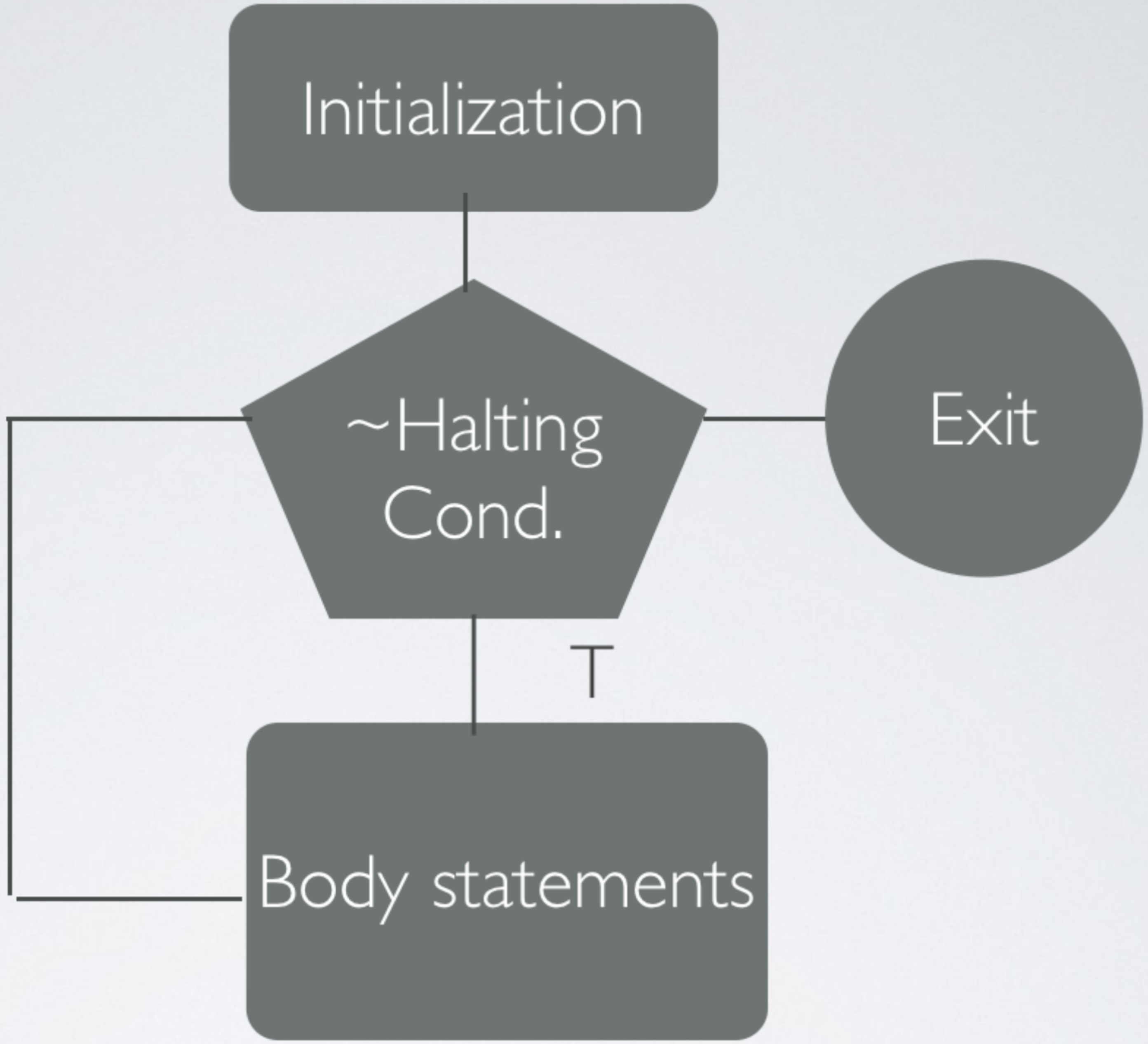


WHILE LOOPING

OUTLINE

- Gene Pattern parsing
- Walking
- Matrix multiplication
- Decimal to binary translation
- Binary search
- Root finding

Matrix multiplication

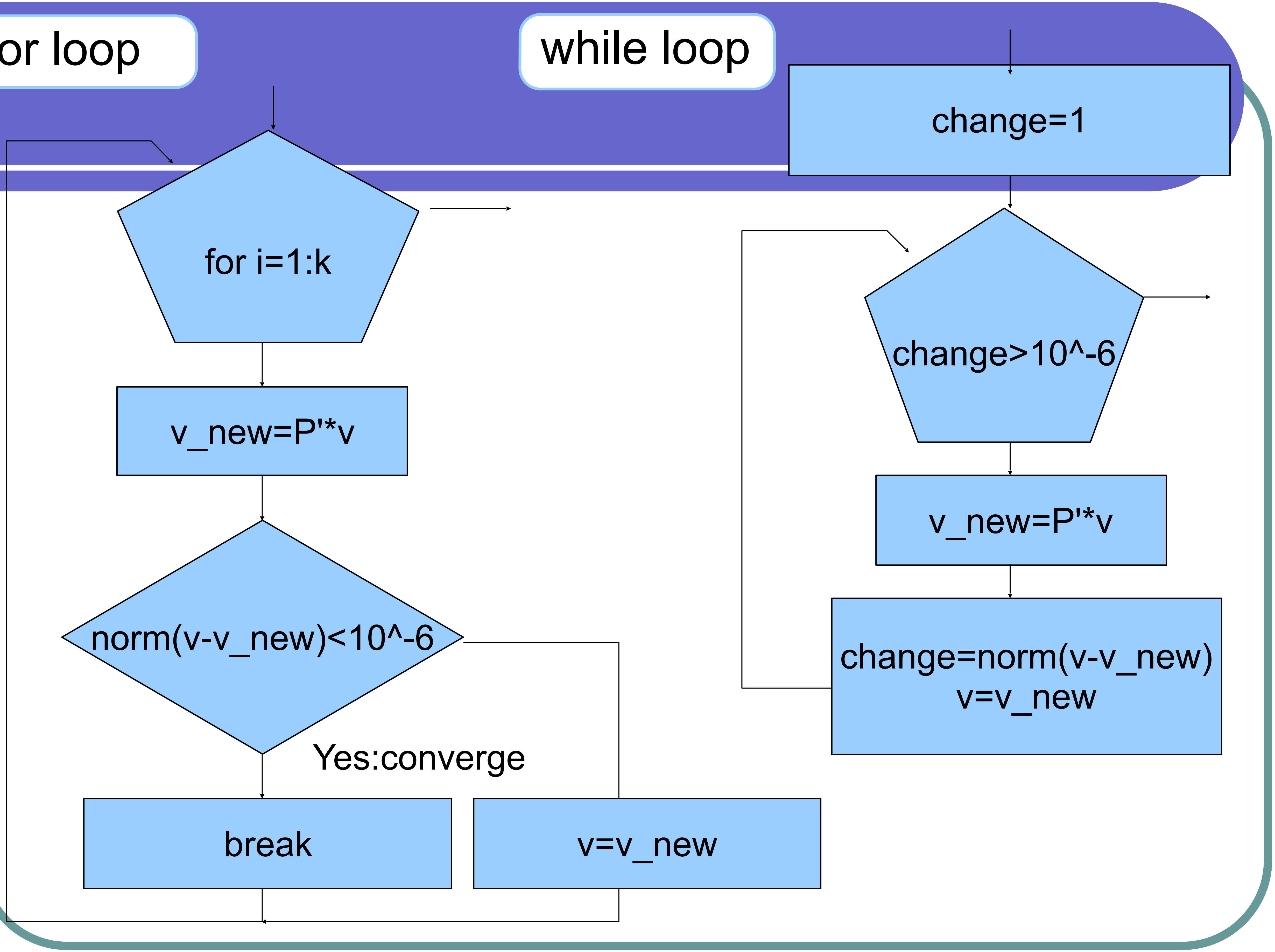
- Draw a flow chart to illustrate finding

$$\lim_{k\to\infty}(\mathbf{P}')^{\mathrm{k}}\,\mathbf{v}$$

by a for-loop

For loop

while loop



for i=1:k

v_new=P'*v

norm(v-v_new)<10^-6

Yes:converge

break

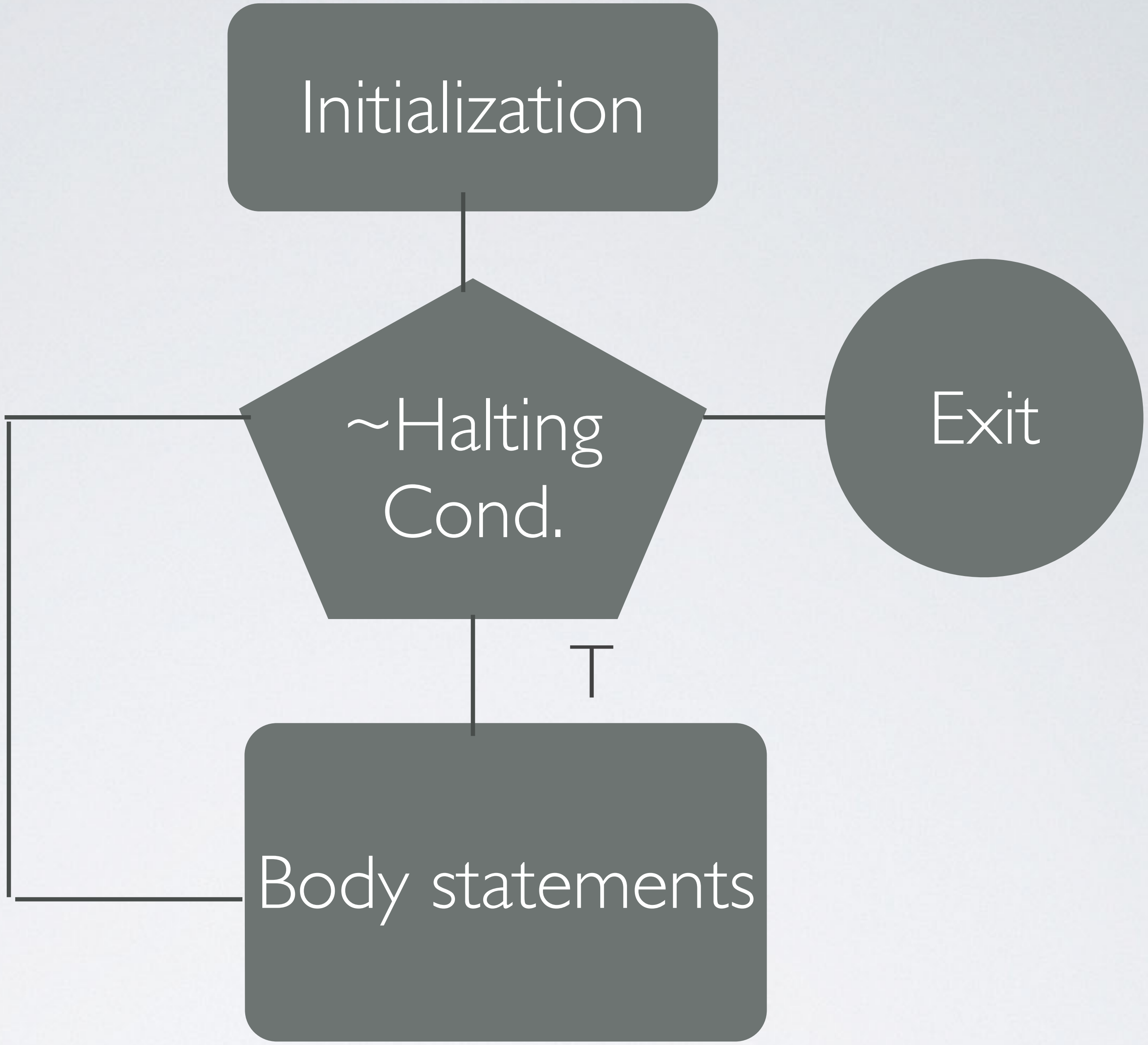
v=v_new

change=1

change>10^-6

v_new=P'*v

change=norm(v-v_new)
v=v_new



- Execute body statements while halting condition is not satisfied

- The entry condition should eventually become false by iteratively executing body statements

WHILE LOOPING

- While-looping uses a halting condition to break iteration
- If the halting condition holds, while-looping breaks
- If the halting condition never holds, while-looping executes endless
- Any programmable program can be implemented in form of while-looping

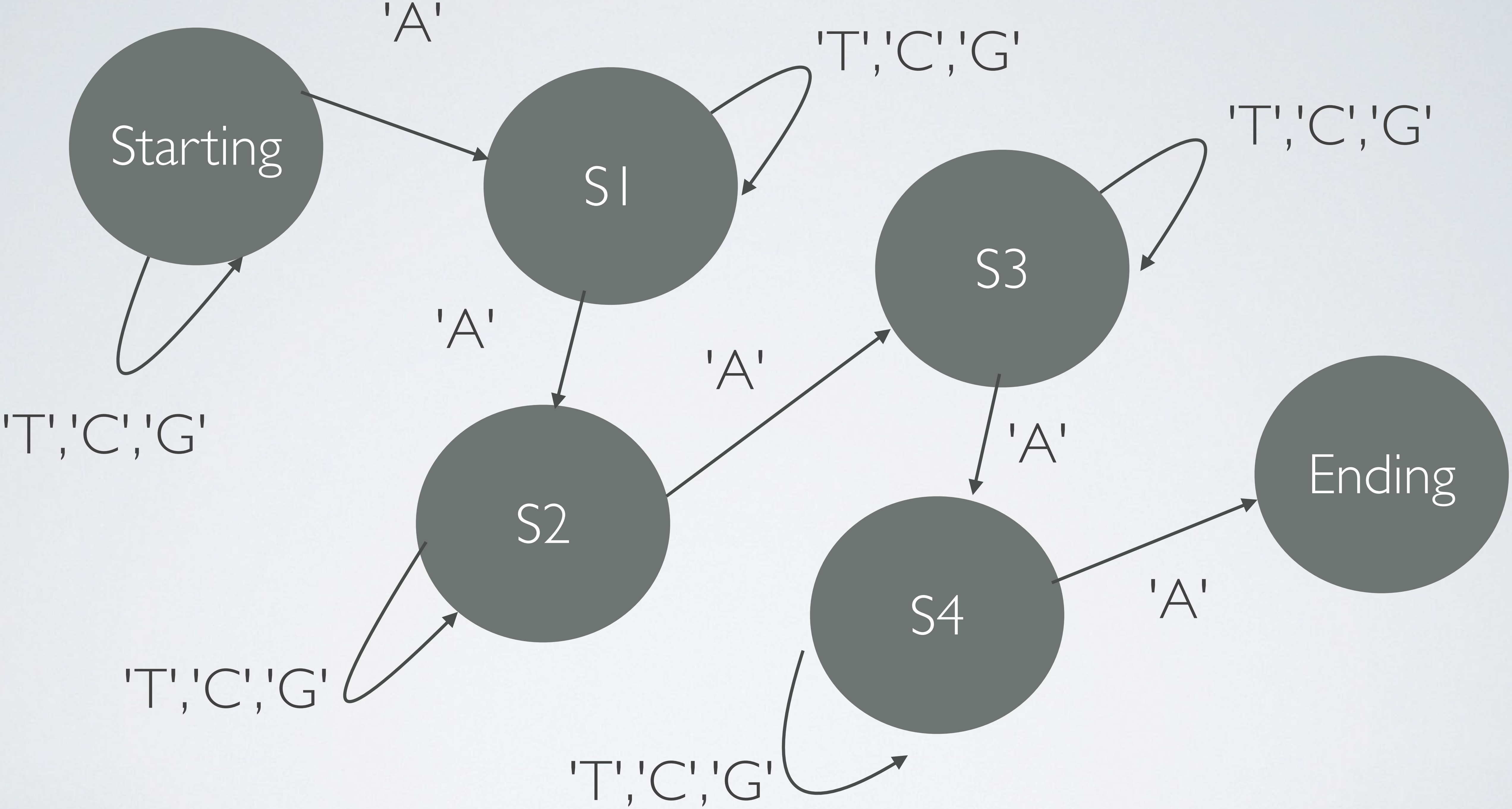
GENE PARSING 5 A'S

- A set of genetic alphabets {A, T, C, G}
- A gene is series of alphabets in {A, T, C, G}
- Draw a while-looping flow chart to get alphabets repeatedly until 5 A's have been given.
- This problem can be implemented by while-looping
- It is unable to determine the length of parsing characters in advance.

A STATE DIAGRAM

- A state diagram is employed to parse a string
- A starting state
- An ending state
- Four internal states respectively represent cumulated A's

5 A'S

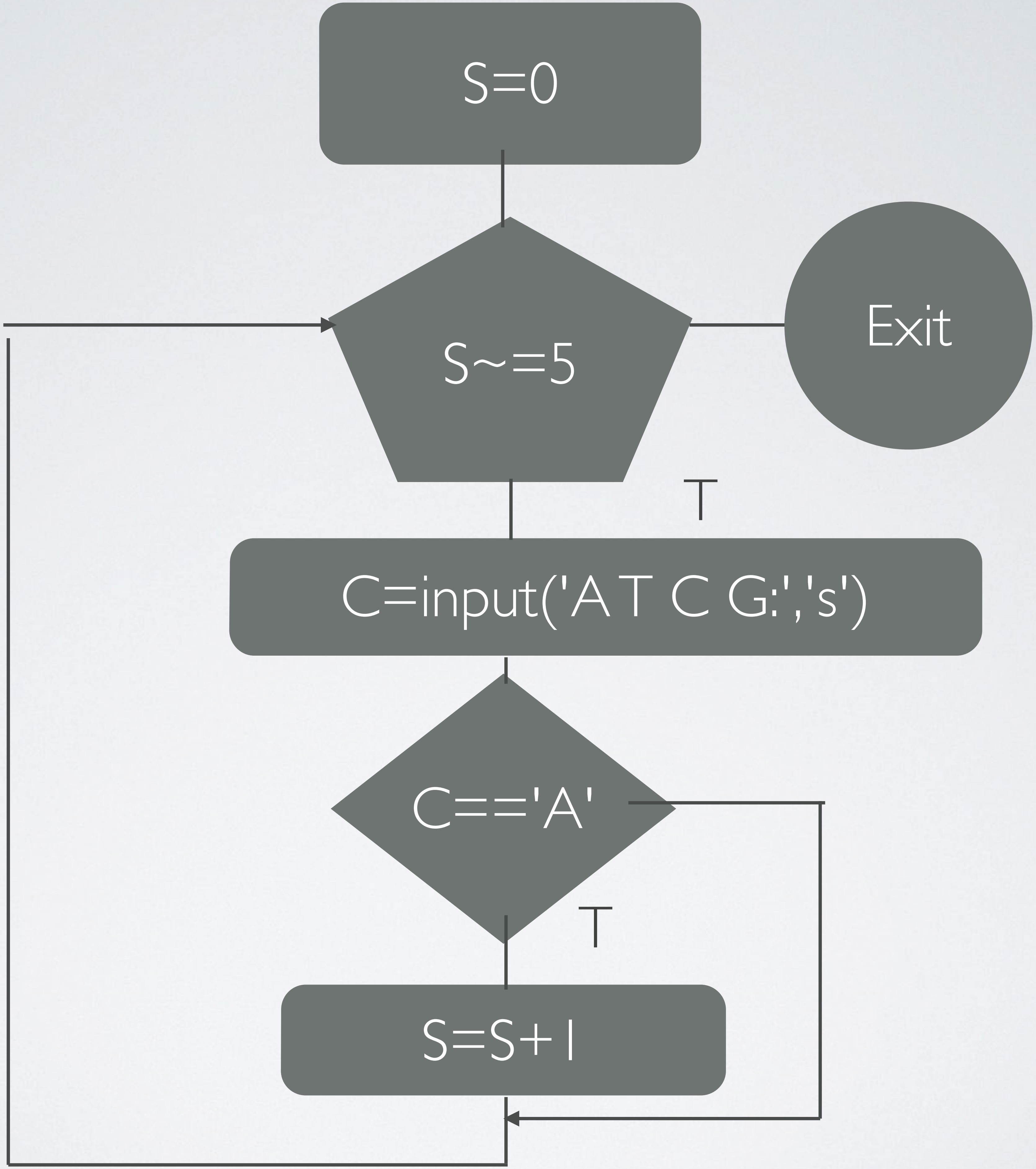


WHILE-LOOPING

- Use an integer to emulate all states
- S is set to zero for initialization
- Increase S by one to emulate state transition whenever 'A' is given
- The halting condition simply checks if S equals 5
- Body statement

  A. Get a character

  B. Update S


S=0
S~=5
Exit
T
C=input('A T C G:','s')
C=='A'
T
S=S+1

PARSING 5 CONSECUTIVE A'S

- A set of genetic alphabets {A, T, C, G}
- A gene is series of alphabets in {A, T, C, G}
- Draw a while-looping flow chart to get alphabets repeatedly until five consecutive A's appear
- This problem can be implemented by while-looping
- It is unable to determine the length of parsing characters.

A STATE DIAGRAM

- State space: s starting state, four internal states, and an ending state

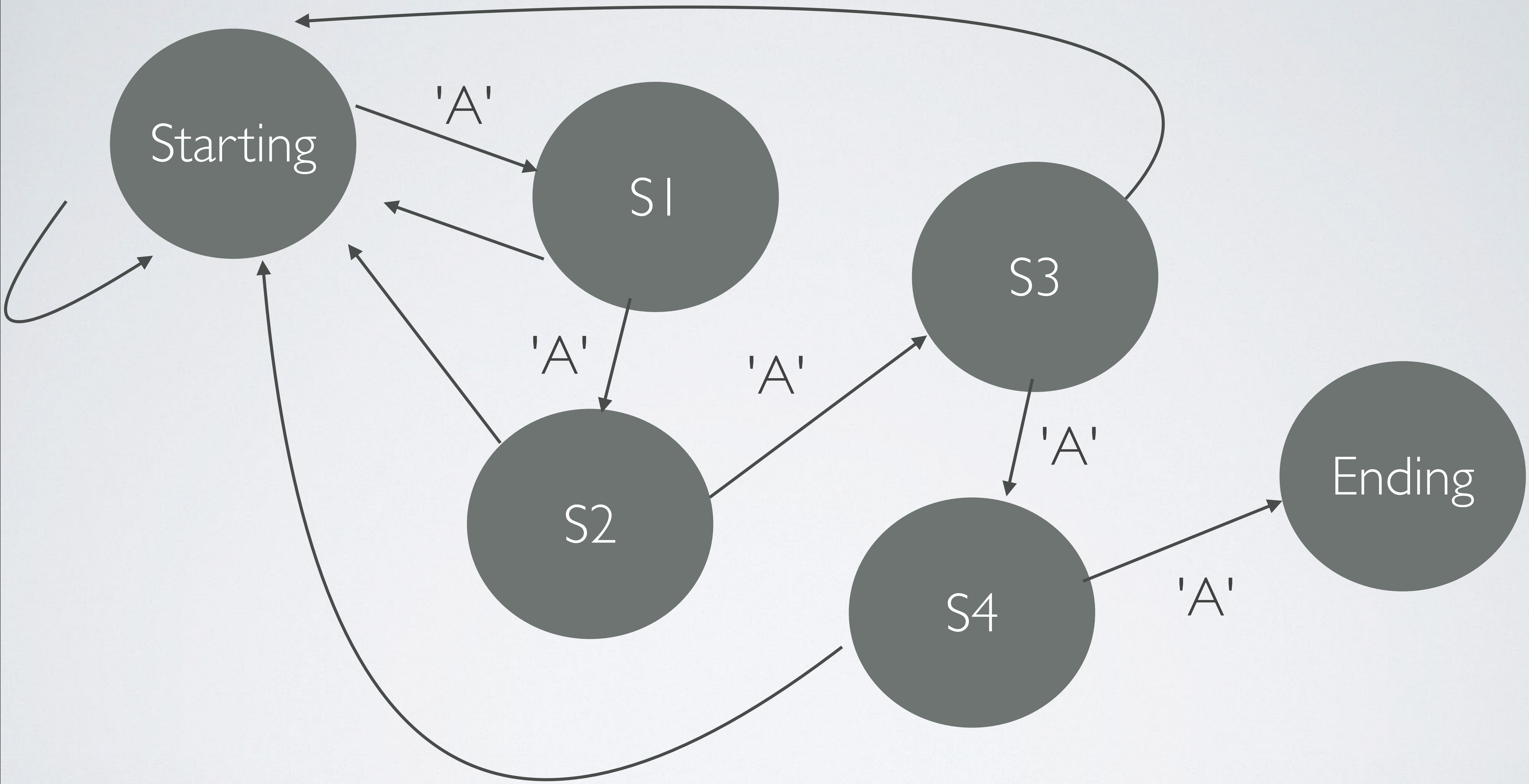
- Four internal states respectively represent the number of consecutive A's

- State transition:

    - if 'A' is given, move to the success state

    - if other characters are given, move to the starting state

5 A'S

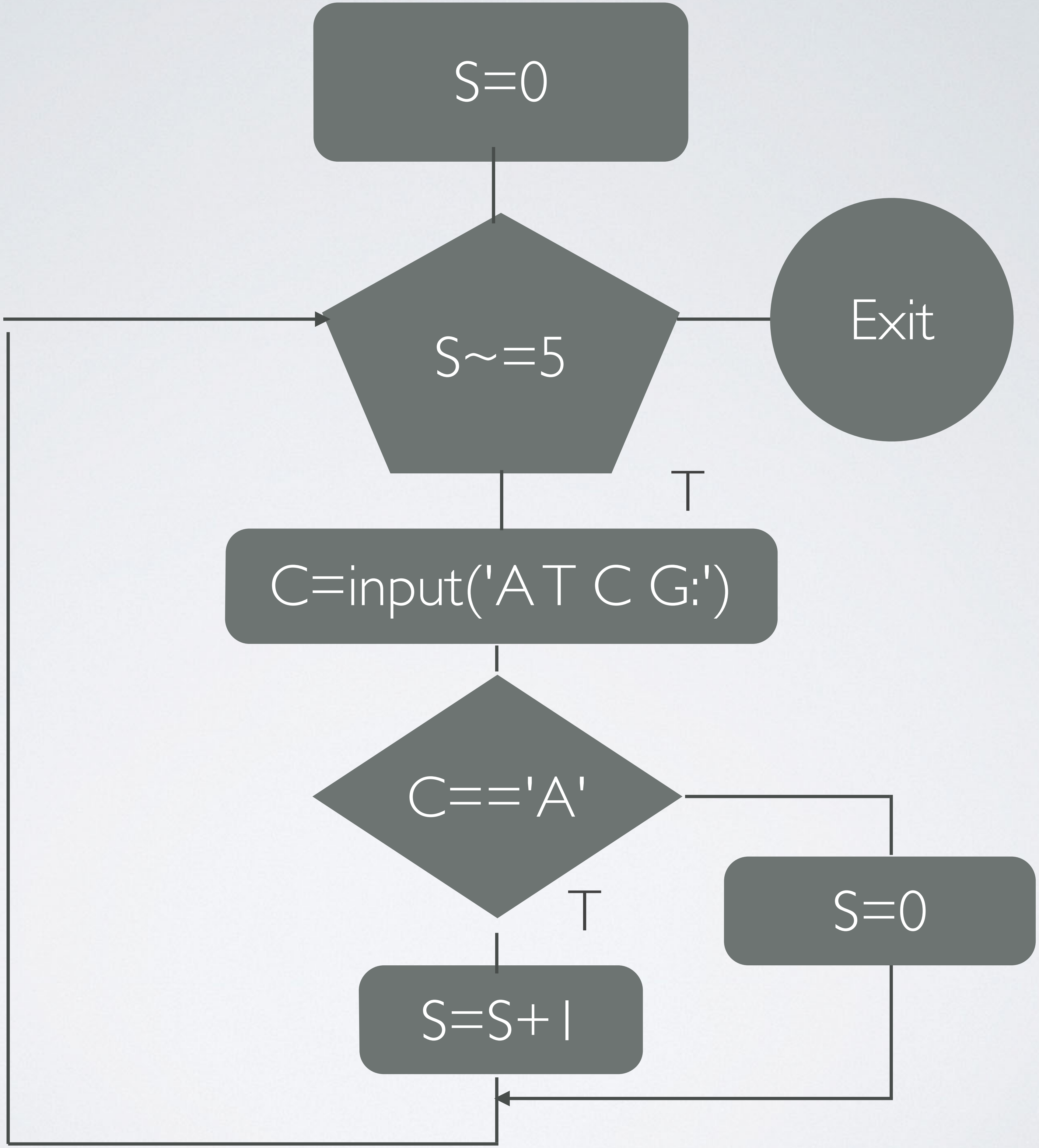


WHILE-LOOPING

- Use an integer to emulate all states
- S is set to zero for initialization
- Increase S by one to emulate state transition whenever 'A' is given
- The halting condition simply checks if S equals 5
- Body statement

    A. Get a character

    B. Update S



S=0

S~=5

Exit

T

C=input('A T C G:')

C=='A'

T

S=0

S=S+1

REPEAT

ATATATATCG

$(A\ T)^{4}CG$

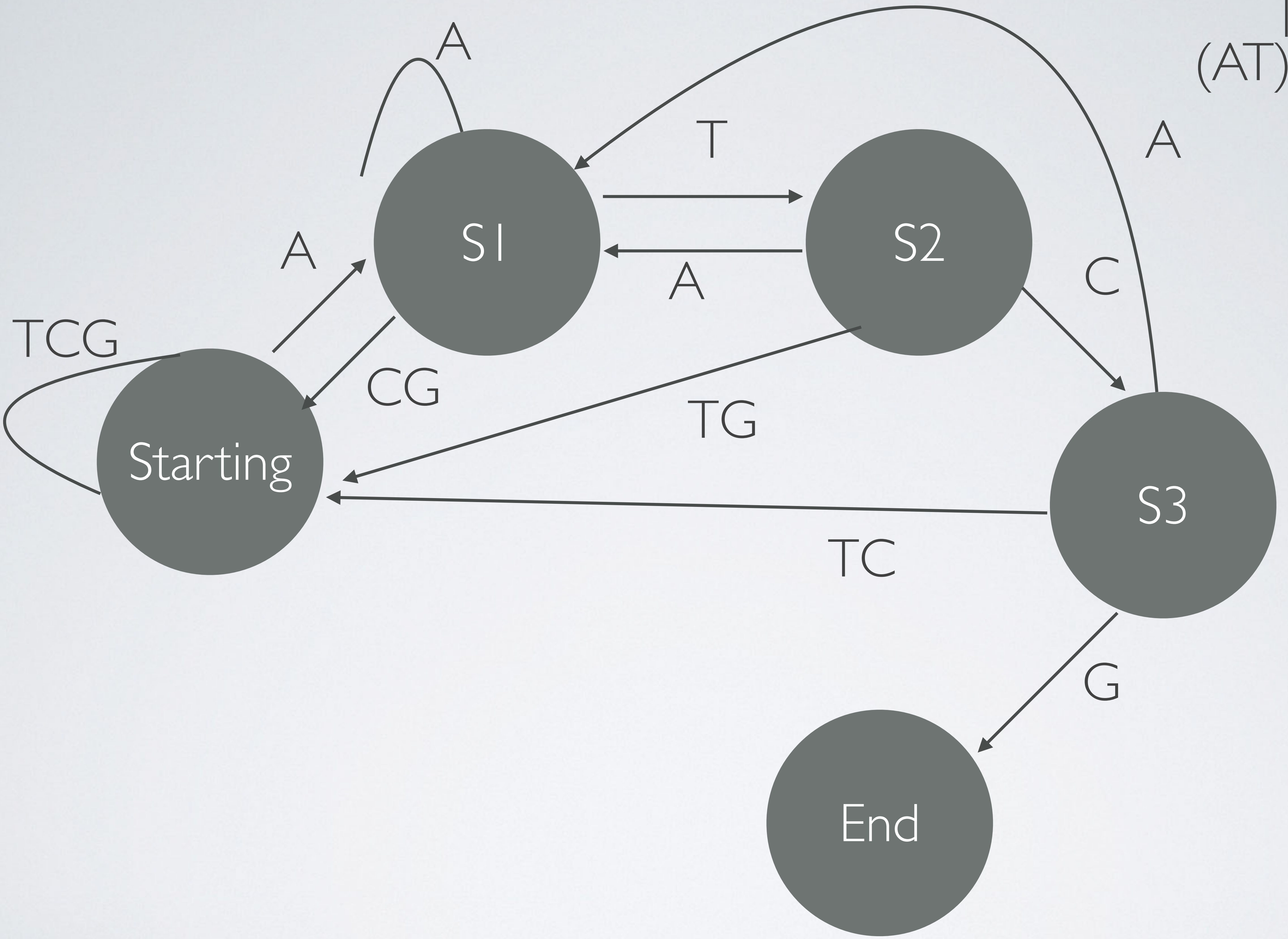
ATATATATATATCG

$(A\ T)^{6}CG$

ATATATATATATGCAAA

$(A\ T)^{6}CG(A)^{3}$

$(AT)^kCG$



PARSING $(AT)^{k}CG$

- A set of genetic alphabets {A, T, C, G}
- A gene is a series of alphabets in {A, T, C, G}
- Draw a while-looping flow chart to get alphabets
- repeatedly until $(AT)^{k}CG$ appear
- This problem can be implemented by while-looping
- It is unable to determine the length of parsing characters.

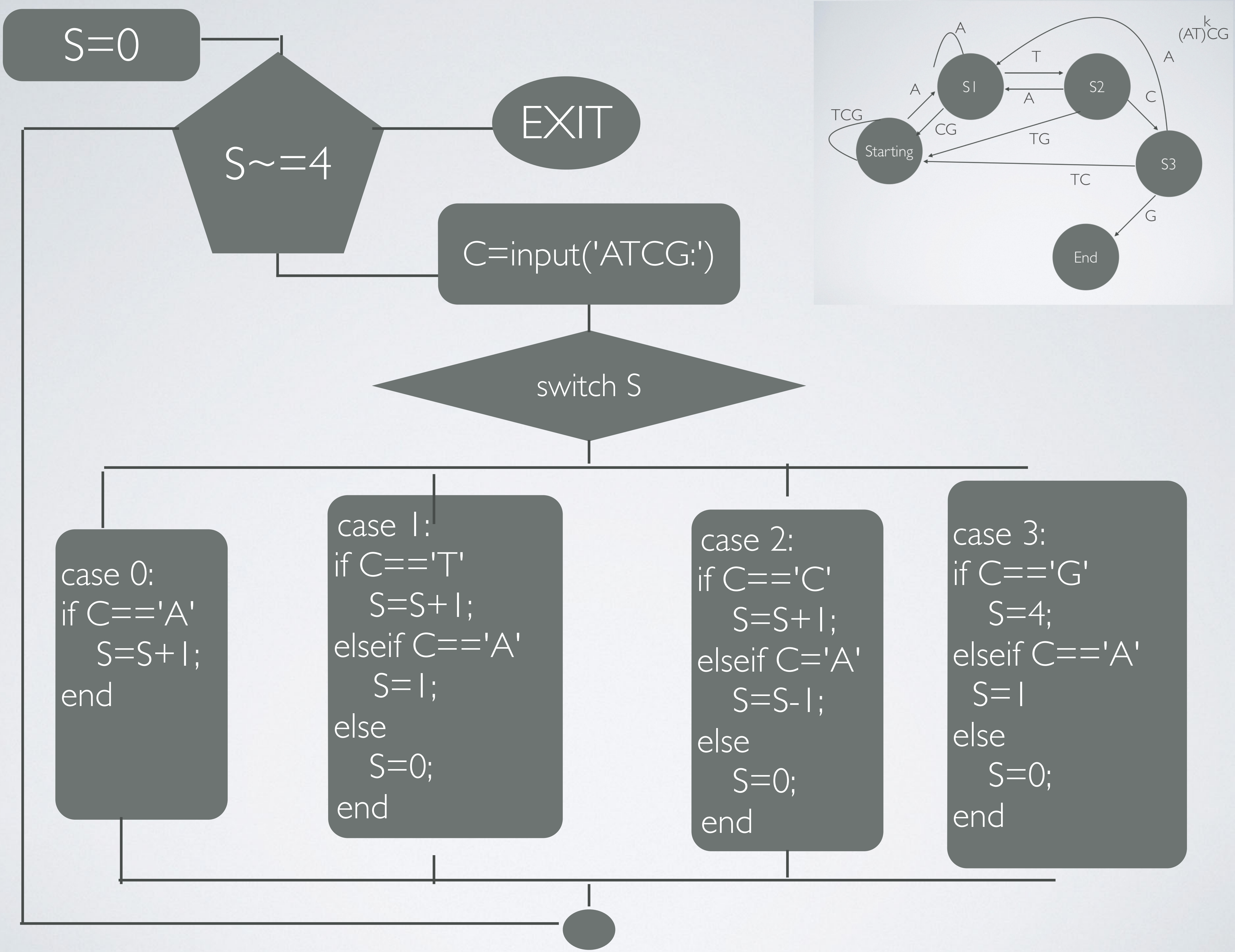
A STATE DIAGRAM

- State space: s starting state, three internal states, and an ending state

- S1: $(AT)^{k}A$, A

- S2: $(AT)^{k}$

- S3: $(AT)^{k}C$



S=0

S~=4

EXIT

C=input('ATCG:')

switch S

```
case 0:
if C=='A'
    S=S+1;
end
```

```
case 1:
if C=='T'
    S=S+1;
elseif C=='A'
    S=1;
else
    S=0;
end
```

```
case 2:
if C=='C'
    S=S+1;
elseif C='A'
    S=S-1;
else
    S=0;
end
```

```
case 3:
if C=='G'
    S=4;
elseif C=='A'
  S=1
else
    S=0;
end
```