Report

From TingTing Shao

Question1

Code:

```
translate([], []).
translate([H|T], [RH|RT]) :-
    codon(H, RH),
    RH \= stop,
    translate(T, RT).
translate([_|_], []).
```

“H” is a variable that represents the first element of the input list in the “translate/2” predicate.

“T” is a variable that represents the rest of the input list, I.e all the elements except the first one.

“RH” is a variable (result of the head of the list) that represent the amino acid corresponding to the codon “H”.

“RT” is a variable that represents the resulting list produced by a recursive call to “translate/2” on the rest of the input list “T”.

Logic and steps to construct the code:

This prolog has three clauses:

- The first clause states that if the list of the codon is empty, return an empty list of amino acids
- The second clause states that if the ‘codon/2’ predicate returns an amino acid that is not “stop” codon, then the first codon in the list is translated and the rest in the list is translated recursively.
- The third clause states if the first codon is a stop codon or the ‘codon/2’ predicate fails, then the list of amino acids is empty.

- **Case1:** with no codon

?-

translate([], X).

X = []

- **Case2:** first one is stop codon

?-

translate([tag,ttt,tct,taa], X).

X = []

- **Case3:** las one is stop codon

?-

translate([atg,ttt,tct,taa], X).

X = [m, f, s]

- **Case4:** after stop codon, there is another codon

?-

translate([atg,ttt,tct,taa, gca], X).

X = [m, f, s]

- **Case5:** with no stop codon

?-

translate([atg,ttt,tct, gca], X).

X = [m, f, s, a]

Question2

Logic and steps to construct the code:

1. Contract a dictionary for elements to map their length
2. Create a function (total_length) to calculate the total length of the terminator
3. Decompose score into three parts:

- Score contributed by link
- Score contributed by elements
- Score contributed by the final two elements: If the final element is PA, and the linker to the end is between 10 and 20 bp, further increase the score by 1

4. Build clauses to calculate the terminator score

- First, check two exceptions: if the total length < 50, or there is no elements, score is 0
- If the two exceptions don't meet, then move on to calculate the score, by summing the scores calculated from element, linker, and final two elements

- **Case1:** with two link > 10bp, three different element, with last one is polya, and the last link next to the last element is longer than 10 bp.

?-

terminator_score([link(10),element(efficiency),link(20),element(positioning),link(10),element(polya),link(12)], Score).

Score = 5.0

- **Case2:** change last element to efficiency

?-

terminator_score([link(10),element(efficiency),link(20),element(positioning),link(10),element(efficiency),link(12)], Score).

Score = 3.0

- **Case3:** change length less than 50

?-

terminator_score([link(10),element(efficiency),link(20)], Score).

Score = 0

- **Case4:** with no elements

?-

terminator_score([link(10),link(20), link(12), link(20), link(10)], Score).

Score = 0

Take case1 as example to show the running trace:

```
//check first exception:
terminator_score(Terminator, 0) :-
    total_length(Terminator, A), A < 50;
//get total length
total_length(Terminator, Length) :-
    total_length(Terminator, 0, Length).
total_length([], Length, Length).
total_length([HIT], Acc, Length) :-
    (   H = element(E)
     ->  lengthMap(E, ElementLength),
        NewAcc is Acc + ElementLength
    ;   H = link(L)
    ->  NewAcc is Acc + L
    ),
    total_length(T, NewAcc, Length).
// determine if length < 50, 82<50,  fail,  check the second exception:
element_count(Terminator, B), B = 0.
//move to element_count predicate
element_count(Terminator, Count) :- element_count(Terminator, 0, Count).
```

```
element_count([], Count, Count).
element_count([H|T], Acc, Count) :-
    (   H = element(_)
    ->  Acc2 is Acc + 1
    ;   Acc2 is Acc
    ),
    element_count(T, Acc2, Count).
//count = 3, 3=0, fail
//move on to terminator_score(Terminator,0,Score). Sequentially check these five conditions, and add all the score together.
check_efficiency(Terminator) :- member(element(efficiency), Terminator).
check_positioning(Terminator) :- member(element(positioning), Terminator).
check_polya(Terminator) :- member(element(polya), Terminator).
link_score
final_two_elements
```