Cohen: Chapter 2

1. Consider the language S*, where S = {a b}.
How many words does this language have of length 2? of length 3? of length n?

Number of words = (Number of letters) $^{(WordLength)}$

Length 2: $2^2 = 4$
Length 3: $2^3 = 8$
Length n: 2^n

2. Consider the language S*, where S = {aa b}.
How many words does this language have of length 4? of length 5? of length 6?
What can be said in general?

Words of length 0: /\ = 1 word
Words of length 1: b = 1 word
Words of length 2:
(Add aa to all words of length 0 → 0 + 2 = 2)
(Add b to all words of length 1 → 1 + 1 = 2)
aa **b**b = 2 words
Words of length 3:
(Add aa to all words of length 1 → 1 + 2 = 3)
(Add b to all words of length 2 → 2 + 1 = 3)
aab **b**aa **b**bb = 3 words
Words of length 4:
(Add b to all words of length 3)
(Add aa to all words of length 2)
baab **b**baa **b**bbb **aa**aa **aa**bb = 5 words
Words of length 5:
(Add aa to the 3 words of length 3)
(Add b to the 5 words of length 4) = 8 words
Words of length 6:
(Add aa to the 5 words of length 4)
(Add b to the 8 words of length 5) = 13 words

In general, words of length n can be formed by adding aa to words of length (n-2) and b to words of length (n-1). This is the Fibonacci sequence!

3. Consider the language S*, where S = {ab ba}. Write out all the words in S* that have seven or fewer letters. Can any word in this language contain the substrings aaa or bbb?
What is the smallest word that is not in this language?

Words of length 0: /\
Words of length 2: ab ba
Words of length 4: **ab**ab **ab**ba **ba**ab **ba**ba
Words of length 6: **ab**abab **ab**abba **ab**baab **ab**baba
baabab **ba**abba **ba**baab **ba**baba

No words can contain aaa or bbb because every a and b is preceded / followed by a different letter, so one letter can never be surrounded on both sides by the same letter.

a and b are the smallest words / strings not in S*.

4. Consider the language S*, where S = {a ab ba}. Is the string (abbba) a word in this language? Write out all the words in this language with six or fewer letters. What is another way in which to describe the words in this language? Be careful, this is not simply the language of all words without bbb.

abba is not in S* because each b must be preceded / followed by a.

Words of length 0:	/\
Words of length 1:	a
Words of length 2:	**a**a ab ba
Words of length 3:	**a**aa **a**ab **a**ba baa
Words of length 4:	**a**aaa **a**aab **a**aba **a**baa **ab**ab **ab**ba **ba**ab **ba**ba **ba**aa
Words of length 5:	**a**aaaa **a**aaab **a**aaba **a**abaa **a**abab **a**abba **a**baab **a**baba **a**baaa **ab**baa **ba**aaa **ba**aab **ba**aba **ba**baa
Words of length 6:	**a**aaaaa **a**aaaab **a**aaaba **a**aabaa **a**aabab **a**aabba **a**abaab **a**ababa **a**abaaa **a**abbaa **a**baaaa **a**baaab **a**baaba **a**babaa **ab**abab **ab**abba **ab**baab **ab**baba **ab**baaa **ba**abab **ba**abba **ba**baab **ba**baba **ba**baaa **ba**aaaa **ba**aaab **ba**aaba **ba**abaa

There are 60 words with 6 or fewer letters.

This is the language of words without bbb, with every bb preceded and followed by an a (i.e. no word can start or end with bb), and with all non-empty strings containing at least one a (so b and bb are not allowed).

5. Consider the language S*, where S = {aa aba baa}. Show that the words aabaa, baaabaaa, and baaaaababaaaa are all in this language. Can any word in this language be interpreted as a string of elements from S in two different ways? Can any word in this language have an odd total number of a's?

aabaa can be factored as (aa)(baa)
baaabaa can be factored as (baa)(aba)(aa)
baaaaababaaaa can be factored as (baa)(aa)(aba)(baa)(aa)

Words can be interpreted in one way only:
* If the word consists of only a's, it can be factored in one way as groups of (aa).
* If there is a single b (or the first b), then
    - if there is an <u>even</u> number of a's (including 0) before the b,
        - that b is the start of (baa) and the a's before it are factored in one way as groups of (aa).
    - if there is an <u>odd</u> number of a's before the b,
        - that b is part of (aba) and the a's before it are factored in one way as groups of (aa).
* If there is a second b, then
    - if the first b was part of (baa), then
        - if there is an <u>even</u> number of a's between the b's, then
            - the second b is part of (baa), and the a's in between are factored in one way as groups of (aa).
        - if there is an <u>odd</u> number of a's between the b's, then
            - the second b is part of aba, and the a's in between are factored in one way as groups of (aa).
    - if the first b was part of (aba), then
        - if there is an <u>even</u> number of a's between the b's, then
            - the second b is part of (aba), and the a's in between are factored in one way as groups of (aa).
        - if there is an <u>odd</u> number of a's between the b's, then
            - the second b is part of (baa), and the a's in between are factored in one way as groups of (aa).

The same argument can be used for words containing more than 2 b's.

No word can have an odd total of a's because all the elements used to build the words have an even number of a's.

6. Consider the language S*, where S = {xx xxx}. In how many ways can x^{19} be written as the product of words in S? This means: How many different factorisations are there of x^{19} into xx and xxx?

x^{19} can consist of 8 doubles (xx) and 1 triple (xxx) → 8*2 + 1*3 = 19
x^{19} can consist of 5 doubles (xx) and 3 triples (xxx) → 5*2 + 3*3 = 19
x^{19} can consist of 2 doubles (xx) and 5 triples (xxx) → 2*2 + 5*3 = 19

3 doubles can be replaced by 2 triples: (xx)(xx)(xx) = (xxx)(xxx)

Let xx = d and xxx = t

The number of ways of factoring 19 x's into d's and t's is equal to:
The number of ways of arranging 8d's and 1t
+ the number of ways of arranging 5 d's and 3 t's
+ the number of ways of arranging 2 d's and 5 t's.

Distinguishable permutations:
If a set of n objects consists of *k different kinds* of objects (with n1 objects of the first kind, n2 objects of the second kind… where n1 + n2 + …nk = n) the number of distinguishable permutations is:
n! / (n1!*n2!*…nk!)
Arrange 8 d's and 1 t: 9!/(8!1!) = 9
Arrange 5 d's and 3 t's: 8!/(5!3!) = 56
Arrange 2 d's and 5 t's: 7!/(2!5!) = 21

9 + 56 + 21 = 86 ways of factoring x^{19} in groups of xx and xxx.

7. Consider the language PALINDROME over the alphabet {a b}. (i) Prove that if x is in PALINDROME, then so is x^n for any n. (ii) Prove that if y^3 is in PALINDROME, then so is y. (iii) Prove that if z^n is in PALINDROME for some n (greater than 0), then z itself is also. (iv) Prove that PALINDROME has as many words of length 4 as it does of length 3. (v) Prove that PALINDROME has as many words of length 2n as it has of length 2n - 1. How many words is that?

(i)
Suppose x has length k: x1 x2 x3 … xk = letters in x.

Since x ∈ PALINDROME, x1 = xk, x2 = x(k-1), x3 = x(k-2), etc…
x^n = (x1 x2…x(k-1) xk)(x1 x2…x(k-1) xk)(x1 x2…x(k-1) xk)… n times.
Because x is in PALINDROME, each x in x^n can be written backwards:
So x^n also = (xk x(k-1)… x2 x1)(xk x(k-1)… x2 x1)(xk x(k-1)…x2 x1)… n times,
which is precisely the same as reverse(x^n).
Since x^n = reverse(x^n), x^n is also a palindrome.

(ii)
Suppose y has length k: y1 y2…y(k-1) yk = letters in y.
Since yyy ∈ PALINDROME, reverse(yyy) = yyy, i.e:
(y1 y2…y(k-1) yk)(y1 y2…y(k-1) yk)(y1 y2…y(k-1) yk) = (yk…y1)(yk…y1)(yk…y1).
This means that: y1 = yk, y2 = y(k-1), etc…
So y ∈ PALINDROME.

(iii)
Suppose z has length k: z1 z2…z(k-1) zk = letters in z.
Since z^n ∈ PALINDROME, reverse (z^n) is also in PALINDROME, i.e:
(z1…zk)(z1…zk)… = (zk…z1)(zk…z1)… (n times).
Therefore, z1 = zk, z2 = z(k-1), etc…
So z ∈ PALINDROME.

(iv)
Since a palindrome is symmetric about the middle, the number of ways of selecting letters for the LHS is the same for the RHS (for every LHS combination, there is a fixed RHS combination).
So, consider the number of ways that a and b can be used for the LHS:
Length(LHS) = 2.
By the product rule, there are 2*2 ways = 4.
(Use the product rule because a and b may be repeated).

So PALINDROME has 4 words of length 4.

A word of length 3 (which is odd) has 1 letter on the LHS and 1 on the RHS (which must be the same), about the middle letter.

Consider the number of ways that a and b can be used to fill the LHS:
length(LHS) = 1,
and there are 2 letters, so there are 2 ways of making up the LHS / RHS.
BUT, because the whole word has odd length, and there is a middle term, we must also consider the number of ways of filling this middle term:
Length(middle term) = 1,
and there are 2 letters, so there are 2 ways of making up the middle term.

By the product rule, 2*2 = 4,
so PALINDROME also has 4 words of length 3.

(v)
Words of length 2n have an even number of letters, with n letters on the LHS (a mirror image of the RHS) and no middle letter.

There are 2^n ways of forming the LHS (and thus the entire word) because there are 2 letters (a and b) that, by the product rule, fill positions 1 → n in 2*2*2… (n times) ways.

Words of length 2n - 1 (odd length) have length n-1 on the LHS & RHS, and a word of length 1 in the middle.
(n-1 + 1 + n-1) = 2n - 1)
Consider the LHS:
There are 2^{n-1} ways of forming the LHS (and thus the RHS too).
There are 2 ways of forming the middle term.
By the product rule, there are therefore $2^{n-1} * 2$ ways of forming the entire word:
$2^{n-1} * 2 = 2^n$ (which is the same number of ways as fords of length 2n).

8. Show that if the concatenation of two words (neither /\) in PALINDROME is also a word in PALINDROME, then both words are powers of some other word; that is, if x and y and xy are all in PALINDROME, then there is a word z such that $x = z^n$ and $y = z^m$ for some integers n and m (maybe n or m = 1).

If x, y ∈ PALINDROME, and so is xy, then if length(x) = length(y), x = y = z.
This is because x1 = xk = yk, x2 = x(k-1) = y(k-1), etc…
If length(x) ≠ length(y):
Let x be the shorter word:
If xy is in PALINDROME, x1 = ym (the first and last letters of xy)

x2 = y(m-1)
xk = y(m - k) (m > k because y is longer than x).
Because y is also in PALINDROME, it follows that
y1 = ym
y2 = y(m-1)
yk = y(m - k + 1)…etc.
Because xy is in PALINDROME, it follows that
y1 = y(m-k)
y2 = y(m - k - 1)
yk = y(m = 2k + 1).
This process can be continued, each time equating k elements of y1 until you've reached the middle. Since you're always using palindromes of length k (the shorter word) at a time, it follows that the longer word must be some power of x. (x might in turn also be a power of a shorter word).

9. (i) Let S = {ab bb} and let T = {ab bb bbbb}. Show that S* = T*.
(ii) Let S = {ab bb} and let T = {ab bb bbb}. Show that S* ≠ T*, but that S* ⊂ T*.
(iii) What principle does this illustrate?

(i)
S* = {/\ and all possible concatenations of ab and bb}
T* = {/\ and all possible concatenations of ab, bb, and bbb}, but bbb is just bb concatenated with itself, so
T* = {/\ and all concatenations of ab and bb} = S*.

(ii)
If one language has the same words as another one, plus additional words made up of concatenations of words already in the language, both those languages have the same closure.

10. How does the situation in Problem 9 change if we replace the operator * with the operator + as defined in this chapter? Note the language S+ means the same as S*, but does not allow the "concatenation of no words" of S.

The situation stays the same because the only change the + operator makes is the elimination of /\ in the closure language.

11. Prove that for all sets S,
(i) (S+)* = (S*)*
(ii) (S+)+ = S+
(iii) Is (S*)+ = (S+)* for all sets S?

(i)
S+ = {all concatenations of words in S, excluding /\}.
(S+)* = {/\ and all concatenations of words in S+}
= {/\ and all concatenations of (concatenations of words in S, excluding /\)}
= {/\ and all concatenations of words in S}
= S*
S* = {/\ and all concatenations of words in S}
(S*)* = {/\ and all concatenations of words in S}
= {/\ and all concatenations of (/\ and all concatenations of words in S)}
= {/\ and all concatenations of words in S}
= S*
Therefore (S+)* = (S*)* = S*

(ii)
S+ = {all concatenations of words in S, excluding /\}
(S+)+ = {all concatenations of words in S+, excluding /\}
= {all concatenations of (all concatenations of words in S, excluding /\) excluding /\)}
= {all concatenations of words in S, excluding /\}
= S+

(iii)
S* = {/\ and all concatenations of words in S}
(S*)+ = {all concatenations of words in S*, but not /\}
S* already contains /\, so (S*)+ contains /\ too, since it's part of the language.
No new words are added with the + operator, so (S*)+ = S*.

S+ = {all concatenations of words in S, without /\}
(S+)* = {/\ and all concatenations of words in S+}
= {/\ and all concatenations of (all concatenations of words in S, not /\)}
= {/\ and all concatenations of words in S}
= S*
The external * operator only added /\ to the language.

Therefore (S*)+ = (S+)* = S*.

12. Let S = {a bb bab abaab}. Is abbabaabab in S*? Is abaabbabbaab? Does any word in S* have an odd total number of b's?

(a)(bb)(abaab)ab can't be factored into substrings from S, so it is not in the language.
(abaab)(bab)b(a)(a)(bb) can't be factored into substrings from S, so it's not in the language.
Neither word is in the language because they both have an odd total of b's.
Words in S* all have an even total of b's because all the elements of S do.

13. Suppose that for some language L we can always concatenate two words in L and get another word in L if and only if the words are *not* the same. That is, for any words w1 and w2 in L where w1 ≠ w2, the word w1w2 is in L but the word w1w1 is not in L. Prove that this cannot happen.

Counter-example:
w1 ∈ L and w2 ∈ L (2 different words)
therefore (w1)(w2) ∈ L
w1w2 ∈ L and w1 ∈ L (2 different words)
therefore (w1w2)(w1) ∈ L
w1w2w1 ∈ L and w2 ∈ L (2 different words)
therefore (w1w2w1)(w2) ∈ L

But (w1w2w1)(w2) can also be factored as: (w1w2)(w1w2), which are 2 equal factors / words, so this language can't exist.

14. Let us define (S**)* = S***
Is this set bigger than S*? Is it bigger than S?

S*** is no bigger than S*. Both sets contain an infinite number of elements.
S*** is bigger than S (if S is not {/\}) because it is made up of all concatenations of elements in S.

15. Let w be a string of letters and let the language T be defined as adding w to the language S. Suppose further that T* = S*.
(i) Is it necessarily true that w ∈ S?
(ii) Is it necessarily true that w ∈ S*?

(i)
No - Not if w is a concatenation of words from S, with this specific concatenation not found in S. (See problem 9iii)
E.g. S = {a b} T = {a b aa} where w = aa
S* = T*, even though w ∉ S.

(ii)
Yes = w ∈ T, so w ∈ T*.
But T* = S*,
so w is also an element of S*.

16. Give an example of a set S such that the language S* has more six-letter words than seven-letter words. Give an example of an S* that has more six-letter words than eight-letter words. Does there exist an S* such that it has more six-letter words than twelve-letter words?

S = {ababab} or {aaa bbb} or {ab bb}.
The length of words in S are all factors of 6. When factors of the same length are concatenated they will never produce a 7-letter word because 7 is prime and greater than 6.

S = {ababab} or S = {aaa}.
S contains a word whose length is a factor of 6 (so there will be a 6-letter word in S*_, but not a factor of 8, so there are no words of length 8 in S*.

There can't be more n-letter words than 2n-letter words because each word of length n can be concatenated with itself (and others) to produce a 2n-letter word. There will always be at least as many 12 than 6-letter words.

17. (i) Consider the language S*, where S = {aa ab ba bb}. Give another description of this language.
(ii) Give an example of a set S such that S* *only* contains all possible strings of a's and b's that have length divisible by 3.
(iii) Let S be all strings of a's and b's with odd length. What is S*?

(i)
The language consisting of any concatenation of a's and b's of even length.

(ii)
If S contains all possible strings of a & b of length 3, then all the words in S* will have length divisible by 3 and will include any concatenation of a's and b's (because S did).
By the product rule, there are 2*2*2 = 8 possible words of length 3:

S = {aaa aab aba baa abb bba bab bbb}

(iii)
S* = {/\ and all possible strings of a's and b's of any length}.

This is because a ∈ S and b ∈ S, and any combination of these letters can result in any word.
Furthermore, an odd word + and odd word = an even word.
an odd word + an odd word + an odd word = an odd word.
So both odd and even words are included.

18. (i) If S = {a b} and T* = S*, prove that T must contain S.
(ii) Find another pair of sets S and T such that if T* = S*, then S ⊂ T.

(i)
S* = {/\ and all concatenations of a's and b's} = T*.
This means that a ∈ T and b ∈ T. (The smallest words in S* and T*).
T may contain other strings (only concatenations of a & b) and still have the same clusure as S*, but it must have at least the elements of s (which are the smallest factors in the definition of S*), so T contains S.

(ii)
T = {a b aa abb} S = {a b aa}
S ⊂ T

19. One student suggested the following algorithm to test a string of a's and b's to see if it is a word in S*, where S = {aa ba aba abaab}:
Step 1, cross off the longest set of characters from the front of the string that is a word in S.
Step 2, repeat step 1 until it is no longer possible.
If what remains is the string /\, the original string was a word in S*. If what remains is not /\ (this means some letters are left, but we cannot find a word in S at the beginning), the original string was not a word in S*. Find a string that disproves this algorithm.

(abaab)aab → 'aab' is not in S, so here the algorithm doesn't work.
(aba)(abaab) is the correct factorisation.

(abaab)a → 'a' is left over, which is not in the language.
(aba)(aba) is the correct factorisation.

In the language S, 'aba' is also part of a longer word, which is why the algorithm doesn't work. (There is ambiguity as to which word the substring 'aba' belongs).

20. A language L1 is smaller than another language L2 if L1 $\subset$ L2 and L1 $\neq$ L2. Let T be any language closed under concatenation; that is, if t1 $\in$ T and t2 $\in$ T, then t1t2 is also an element of T. show that if T contains S but T $\neq$ S*, then S* is smaller than T. We can summarise this by saying that S* is the smallest closed language containing S.

If T contains S, that means T contains all concatenations of words in S, i.e. S* $\subseteq$ T* (by the definition of T).
T may also have other elements not found in S, which are concatenated with each other and elements of S.
If this is the case, then T $\neq$ S*, but T is greater than S*.