SCHOOL of COMPUTING
SKOOL vir REKENAARKUNDE

THEORETICAL COMPUTER SCIENCE 3
TEORETIESE REKENAARWETENSKAP 3

TUTORIAL LETTER 202
STUDIEBRIEF 202

COS301Y

Discussion of Assignment 02
Bespreking van Werkopdrag 02

1 AFRIKAANSE STUDENTE

Hierdie modeloplossing word net in Engels uitgestuur. As u enige probleem ondervind met die taal, kontak ons asseblief. Onthou ook dat u die werkopdrag-oplossings en later die eksamen in Afrikaans en/of Engels kan skryf al het u die ander taal by registrasie verkies.

2 STUDY MATERIAL

By this time you should have received the following tutorial matter. If you have not received all of these. Please contact the Department of Despatch at the telephone number given in the inventory you received upon registration. The material may also be downloaded from *myUnisa* or *osprey*—see Tutorial Letter COSALLF/301/4/2009.

COSALLF/301/4/2009 General information about the School and study at Unisa.
COSALLF/302/4/2009 Names and telephone numbers of lecturers.
COS301Y/101/0/2009 General information about the module and the assignments.
COS301Y/102/0/2009 Solutions to additional exercises.
COS301Y/201/0/2009 Solutions to Assignment 01.
COS301Y/202/0/2009 This tutorial letter.
COS301Y/501/0/2009 Tutorial letter acting as Study Guide.

Please do NOT contact the School about missing tutorial matter, cancellation of a module, payments, enquiries about the registration, and so on but rather the various relevant administrative departments as indicated in *Unisa: Services and Procedures*. The general Unisa Contact Centre numbers are

0861 670 411 (South Africa) or
+27 11 670 9000 (International)

The School should only be contacted about academic matters. Always use your student number when you contact the University.

3 ADMISSION TO THE EXAMINATION

In order to get examination admission, you had to submit Assignment 01 before its due date.

4 YEAR MARKS

The mark that you obtain for every assignment will contribute to a year mark which, in its turn, contributes 10% towards you final mark for this module. All four assignments carry a weight of 25% each towards this year mark. For example, suppose a student gets 60% for Assignment 01, 70% for Assignment 02, 40% for Assignment 03 and 50% for Assignment 04. Then, the year mark will be:

(0.60 x 0.25 x 10) + (0.70 x 0.25 x 10) + (0.40 x 0.25 x 10) + (0.50 x 0.25 x 10)
= 1.50 + 1.75 + 1.00 + 1.25
= 5.50

The marks obtained in the examination will form 90% of the final mark. (However, the year mark will not play any role in the supplementary examination.)

5 COS301Y: SOLUTIONS TO ASSIGNMENT 02

As stated in Tutorial Letter COS301Y/101/0/2009, we did not mark all the questions, so **please work carefully through the solutions and compare our answers to yours**. Also remember to read our comments on your effort with care.

In all questions except Question 1, we assume that the alphabet is $\Sigma = \{a, b\}$.

5.1 QUESTION 1 (problem 16, page 375 of *Cohen*)

In order to answer this question we use the pumping lemma with length. We proceed as on pages 15–17 of Tutorial Letter COS301Y/501/0/2009, the "Study Guide".

The alphabet is $\Sigma = \{a, b, c\}$ and every word in the language VERYEQUAL consists of the same number of a's, b's and c's, and they can be arranged in any order.

- The first step is to assume that the language

  $$\text{VERYEQUAL} = \{\text{all strings with an equal number of a's, b's, c's}\}$$

  actually is context-free. This means that there exists a context-free grammar (CFG) in Chomsky normal form (CNF) with, say, p live productions which generates the language. Because we assume that the language is context-free, we may apply the pumping lemma. We are going to use the pumping lemma with length. To save space, we call VERYEQUAL now L.

  According to the pumping lemma with length any word w in L with more than 2^p characters can be broken up into five parts, i.e. the word can be written as $w = uvxyz$, with

  - $\text{length}(vxy) \leq 2^p$ and
  - $\text{length}(x) > 0$ and
  - $\text{length}(v) + \text{length}(y) > 0$,
  - and where all words of the form uv^nxy^nz with $n > 1$ are also in the language.

- Now we must choose a suitable word from L which is long enough. The word $a^{2^p}b^{2^p}c^{2^p}$ seems to be a good choice: It is in L and it has more than 2^p characters.

  Let us now examine the different ways in which the word $a^{2^p}b^{2^p}c^{2^p}$ can be broken up into five parts. Remember that vxy cannot have more than 2^p characters. This means that there are five possible ways in which vxy may occur within the word: (i) fully from the first 2^p characters (thus consisting of a's only), (ii) covering some of the a's and also some of the b's, (iii) fully from the group of b's, (iv) covering some of the b's and also some of the c's, and (v) fully from the group of c's. We will look at each of these five possibilities.

1. Suppose vxy consists entirely of a's.

   According to the pumping lemma with length, the word $uvvxyyz$ is also in the language L. This word will have **more than** 2^p a's at the beginning, still followed by $b^{2^p}c^{2^p}$. Such a word is, however, **not** in L.

2. Suppose vxy consists of a's followed by b's.
   According to the pumping lemma with length, the word $uvvxyyz$ is also in the language L. This pumped word will have **more than** 2^p a's at the beginning and/or **more than** 2^p b's with still only 2^p c's at the end. Thus, the pumped word will **not** be in L.

3. Suppose vxy consists entirely of b's.
   According to the pumping lemma with length, the word $uvvxyyz$ is also in the language L. This word will now have 2^p a's at the beginning, followed by **more than** 2^p b's and then followed by c^{2^p}. Such a word is **not** in L.

4. Suppose vxy consists of b's followed by c's.
   According to the pumping lemma with length, the word $uvvxyyz$ is also in the language L. This pumped word will have 2^p a's at the beginning, followed by **more than** 2^p b's and/or **more than** 2^p c's. Thus, the pumped word will **not** be in L.

5. Suppose vxy consists entirely of c's.
   According to the pumping lemma with length, the word $uvvxyyz$ is then also in the language L. However, this word will have **more than** 2^p c's at the end, preceded by $a^{2^p}b^{2^p}$. Such a word is **not** in L.

- We have now seen that all five possible choices of vxy lead to the fact that the word uv^2xy^2z cannot be in the language L. This contradicts the pumping lemma with length. This means that the initial assumption that L is context-free, must be discarded (because if it were context-free, each pumped word would also have been in the language according to the pumping lemma). We conclude that the given language L is **not** context-free.

This concludes the proof.

- Please note again that the choice of a suitable word is very important:

On page 16 of Tutorial Letter COS301Y/501/0/2009 we give an example of a word that is *not* suitable for pumping (when we want to apply the pumping lemma with length). In addition, we actually could not find *any* suitable word in VERYEQUAL where the pumping lemma without length can be applied!

- Note in the above we work with 2^p and NOT with $2p$.

5.2 QUESTION 2 (problem 2(iv), page 398 of *Cohen*)

One possible CFG which generates the language is:
$S \longrightarrow ABC$
$A \longrightarrow aAb \mid ab$
$B \longrightarrow bBa \mid ba$
$C \longrightarrow aCb \mid ab$

We used the given hint to construct this grammar. The production $A \longrightarrow aAb$ can be applied $p-1$ times followed by the production $A \longrightarrow ab$ to produce p a's and p b's. Similarly, q b's and q a's can be produced from the nonterminal B by applying the production $B \longrightarrow bBa$ a total of $q-1$ times, followed by the production $B \longrightarrow ba$. In a similar way, the nonterminal C can produce r a's and r b's. Note that the shortest word is *abbaab*.

5.3 QUESTION 3 (problems 5(i) and 5(ii), page 399 of *Cohen*)

5.3.1 (i)

One possible CFG for this language is
$S \longrightarrow aB$
$B \longrightarrow bbB \mid \Lambda$

5.3.2 (ii)

The language given in this question is the closure of the language generated by the grammar in **(i)**. We follow the algorithm given in the proof of Theorem 38 where a grammar is constructed for the closure of a language which is generated by a given grammar. We have to change all occurrences of the nonterminal S in the given grammar to the nonterminal S_1 and add the two new productions $S \longrightarrow S_1S \mid \Lambda$. This means that in our case the required grammar will be:
$S \longrightarrow S_1S \mid \Lambda$
$S_1 \longrightarrow aB$
$B \longrightarrow bbB \mid \Lambda$

5.4 QUESTION 4 (problem 7(ii), page 399 of *Cohen*)

We are dealing with the language $\{a^xb^ya^z$,where $x+z=y$ and $x \geq 1$, $y \geq 1$ and $z \geq 1\}$. As you can see, all the permissible words have one or more a's, followed by one or more b's, and followed by one or more a's. The number of b's should be equal to the total of all the a's. In Figure 1, we give a correct deterministic pushdown automaton (PDA) that will accept this language. The idea is to "count" the a's at the beginning of the word by pushing an X onto the stack for every a that is read. The stack will contain x X's when all the a's have been read. When we read the b's, we pop the stack for every b that is read until the stack becomes empty. At this point we have then read the first x b's (actually $x+1$). Now we have to "count" the second batch of b's so that we can check the number against the number of a's that will follow them. We do this by pushing an X onto the stack (which was empty) for every b that is read. When we encounter the first a, the stack will contain $(y-x)$ a's. Then, we pop the stack for every a until the stack becomes empty. At this point, the input should also be finished. Let us see how this is implemented in the given PDA:

In READ_0 the first a is read and in PUSH_1 an X is pushed onto the stack to ensure that at least one a is read and counted. The READ_1-PUSH_1-READ_1 cycle "counts" the rest of the x a's at the beginning of the word by pushing an X onto the stack for every a. The POP_1-READ_2-POP_1 cycle "counts" the first batch of b's until the stack becomes empty (when we know we have now read x b's). The PUSH_2-READ_3-PUSH_2 cycle "counts" the rest of the b's by pushing an X onto the stack for every b. There are $(y-x)$ of these X's on the stack when all the b's have been read. The POP_2-READ_4-POP_2 cycle "counts" the z a's at the end. Because z should be equal to $(y-x)$, the stack should be empty when the input is finished.

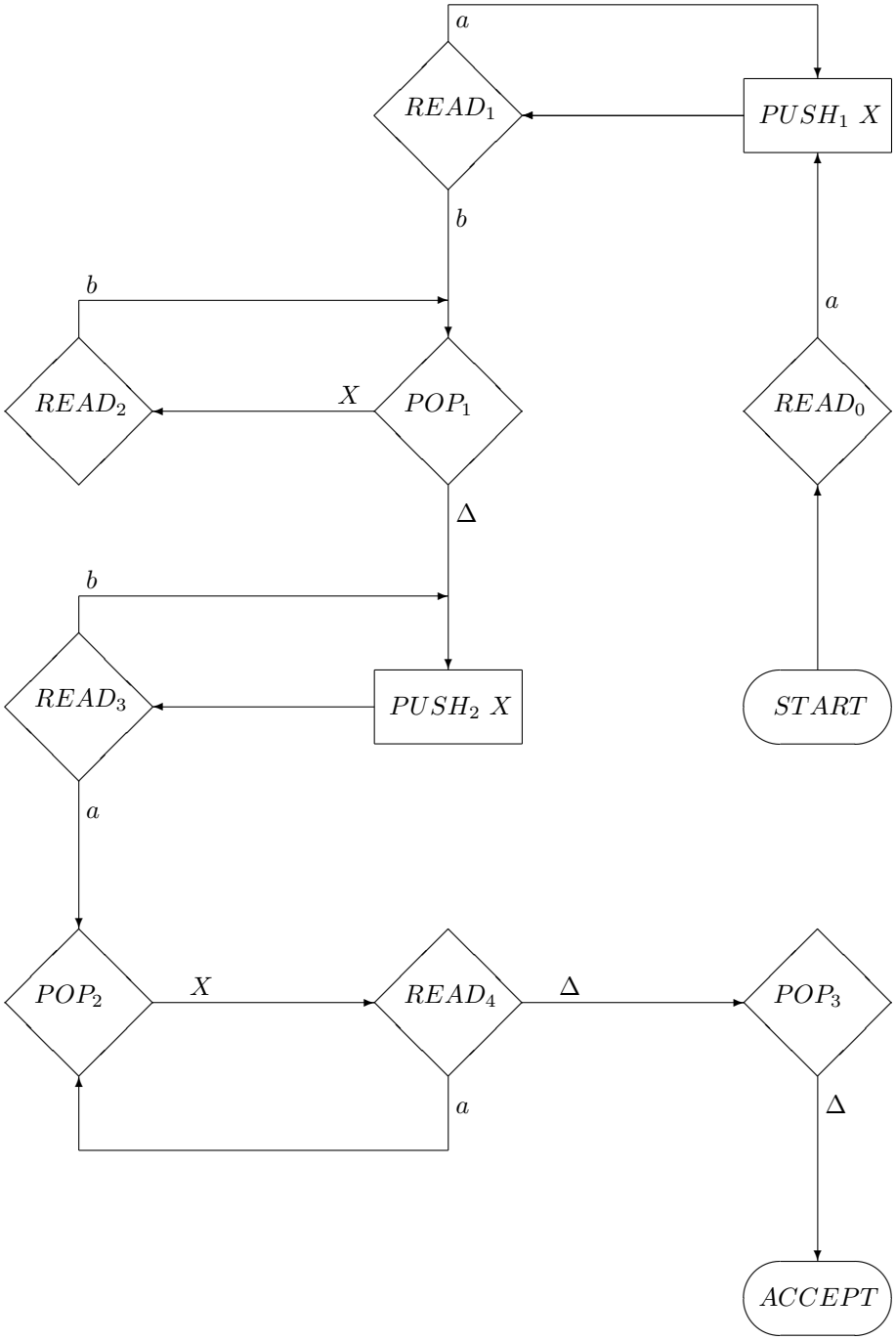

Figure 1: PDA (Question 4)

We hope that you agree that the given machine will accept all the words of the language. Is it possible that it will also accept some impermissible words? Let us look at a few impermissible cases:

- Words starting with b: crashes in READ_0
- Words without b's: crashes in READ_1 (when Δ is read)
- Words without a second group of a's: crashes in READ_3 (when Δ is read)
- Words with a second group of b's: crashes in READ_4
- Words with $x + z \neq y$: crashes in POP_2 (when Δ is popped because $z > y - x$) or POP_3 (when X is popped because $z < y - x$).

5.5 QUESTION 5 (problem 1(iii), page 429 of *Cohen*)

We have to use the algorithm of Theorem 42 to decide whether the given grammar generates any words, i.e. whether the language is empty or not. The theorem is summarised on pages 20–21 of Tutorial Letter COS301Y/501/0/2009 (your "Study Guide").

- Step -1: We have to find out whether S is nullable. We see that it is *not* the case.
- Step 0: We need to convert the CFG to CNF. The given grammar is already in CNF.
- Step 1: We see that the nonterminal D has a production of the required form, namely

  $D \longrightarrow a$,

  and thus can be eliminated. There is no other production with D on the left-hand side, but we do replace all occurrences of D on the right-hand side with a. Now the grammar looks as follows:

  $S \longrightarrow AB$

  $A \longrightarrow BC$

  $C \longrightarrow aA$

  $B \longrightarrow Ca$

  $A \longrightarrow b$
- Step 2: Has S been eliminated? We see that S has not been eliminated yet.
- Step 3: Can another nonterminal be eliminated by performing step 1? The nonterminal A can be eliminated, so we move to step 1.
- Step 1: The nonterminal A has a production of the required form, namely

  $A \longrightarrow b$,

  and thus can be eliminated. The production $A \longrightarrow BC$ has A on the left-hand side and is thrown out. We replace all occurrences of A on the right-hand side with b. The grammar now looks as follows:

  $S \longrightarrow bB$

  $C \longrightarrow ab$

  $B \longrightarrow Ca$
- Step 2: S has not been eliminated yet.
- Step 3: The nonterminal C can be eliminated, so we move to step 1.
- Step 1: The nonterminal C has a production of the required form, namely

  $C \longrightarrow ab$,

  and thus can be eliminated. There is no other production with C on the left-hand side, but we do replace all occurrences of C on the right-hand side with ab. The grammar now looks as follows:

  $S \longrightarrow bB$

  $B \longrightarrow aba$

- Step 2: S has not been eliminated yet.
- Step 3: The nonterminal B can be eliminated, so we move to step 1 again.
- Step 1: The nonterminal B has a production of the required form, namely

  $B \longrightarrow aba$,

  and thus can be eliminated. There is no other production with B on the left-hand side, but we do replace all occurrences of B on the right-hand side with aba. The grammar now looks as follows:

  $S \longrightarrow baba$
- Step 2: S has not been eliminated yet.
- Step 3: The nonterminal S can be eliminated, so we move to step 1 again.
- Step 1: S is eliminated.
- Step 2: S has been eliminated. This means that the language generated by the grammar, is not empty, i.e. the given grammar does generate words.

5.6 QUESTION 6 (problem 3(ii), page 429 of *Cohen*)

We have to decide whether the language generated by the given grammar is finite or infinite by using Theorem 44. This algorithm is summarised on page 409 of *Cohen*.

- **Step 1 of Theorem 44**: In this step all nonterminals which are not used to produce any word should be eliminated. We have to apply the algorithm of Theorem 43 on page 407 of *Cohen*, which demands, in turn, the application of the algorithm of Theorem 42. So, we first apply the algorithm summarised on pages 20–21 of the Study Guide, to the given grammar. Note that we, of course, modify the algorithm slightly, because we do not only want to establish whether the language is empty but rather want to identify all nonterminals leading to no words. For example, step 2 on page 20 is not executed.

  1. *Page 20 of the Study Guide, step -1*: No nonterminal is nullable.
  2. *Page 20 of the Study Guide, step 0*: Although one production is not in CNF (namely $X \longrightarrow ab$), we need not convert it to CNF because the algorithm makes provision for productions with more than one terminal on the righthand side.
  3. *Page 20 of the Study Guide, step 1*: There are two nonterminals N with a production of the form $N \longrightarrow t$ where t is a terminal or a string of terminals, namely $S \longrightarrow b$ and $X \longrightarrow ab$. We therefore know that these nonterminals (namely S and X) are productive and may be useful. Let us start with the nonterminal S. We have to eliminate all productions having S on the left-hand side, i.e. $S \longrightarrow XS$ and $S \longrightarrow b$. Then, we have to replace S with b whenever S occurs on the right-hand side of a production, but there is none. We are left with the grammar below and then go to step 3.

     $X \longrightarrow YZ$

     $Z \longrightarrow XY$

     $X \longrightarrow ab$
  4. *Page 20 of the Study Guide, step 3*: There is one nonterminal N with a production of the form $N \longrightarrow t$, where t is a terminal or a string of terminals, namely $X \longrightarrow ab$. Therefore, we know that this nonterminal (namely X) is productive and may be useful. Go to step 1.
  5. *Page 20 of the Study Guide, step 1*: We eliminate all productions having X on the left-hand side. Then, we replace X with ab whenever X occurs on the right-hand side of a production. We are left with the grammar below and then go to step 3.

     $Z \longrightarrow abY$
  6. *Page 20 of the Study Guide, step 3*: There is no nonterminal N left with a production of the form $N \longrightarrow t$, where t is a terminal or a string of terminals. The remaining production namely $Y \longrightarrow abY$ does not have this form. Therefore, we know that this nonterminal (namely Y) is unproductive.
  7. *Step 1 of the algorithm of Theorem 43* is now completed. There is one unproductive nonterminal.

8. *Step 2 of the algorithm of Theorem 43*: We eliminate all productions involving the unproductive nonterminal Y. Thus, the grammar looks as follows:
   $S \longrightarrow XS \mid b$
   $X \longrightarrow ab$
9. Now *steps 3–6 of the algorithm of Theorem 43* should be executed in order to determine which of the nonterminals S and X are actually used in the generation of words. However, it is unnecessary to execute the steps for S because if we paint the S's in step 3, step 6 follows immediately. So it is clear that S is useful (used in the production of words). Let us investigate X.
10. *Step 3 of the algorithm of Theorem 43 with* X (in the algorithm) $= X$ (in the grammar). We get
    $S \longrightarrow \overset{blue}{X}\, S \mid b$
    $\overset{blue}{X} \longrightarrow ab$
11. *Steps 4–5 of the algorithm of Theorem 43 with* X (in the algorithm) $= X$ (in the grammar). We see that S becomes blue. We are left with
    $\overset{blue}{S} \longrightarrow \overset{blue}{X}\,\overset{blue}{S} \mid b$
    $\overset{blue}{X} \longrightarrow ab$
12. *Step 6 of the algorithm of Theorem 43 with* X (in the algorithm) $= X$ (in the grammar). Because S is blue, we know that X is a useful member of the grammar.
13. At this point we have examined all the nonterminals for usefulness.
14. At last we are able to finish **step 1 of the algorithm of Theorem 44** on page 409 of *Cohen* and then proceed to step 2. Up to now, we have established that two nonterminals, namely S and X, are useful. We are left with the following grammar:
    $S \longrightarrow XS \mid b$
    $X \longrightarrow ab$

- **Step 2 of the algorithm of Theorem 44**: Let us choose the nonterminal S and check whether it is self-embedded, thus X (in the algorithm) $= S$ (in the grammar):

  1. *Step (i)*: We use Ξ instead of the Russian letter. Replace S with Ξ where it appears on the left-hand side of a production:
     $\Xi \longrightarrow XS \mid b$
     $X \longrightarrow ab$
  2. *Step (ii):*
     $\Xi \longrightarrow X\,\overset{blue}{S} \mid b$
     $X \longrightarrow ab$
  3. *Step (iii):*
     $\overset{blue}{\Xi} \longrightarrow X\,\overset{blue}{S} \mid b$
     $X \longrightarrow ab$
  4. *Step (iv):*
     Nothing new can be painted blue.
  5. *Step (v):* Ξ is painted blue, thus S is self-embedded.

- **Step 3 of the algorithm of Theorem 44**: We have finally completed our investigation. Because a useful nonterminal is self-embedded, the language generated by the given grammar, is **infinite**.

5.7 QUESTION 7(problem 9, page 430 of *Cohen*)

We follow the steps of the CYK algorithm as formulated on page 21 of Tutorial Letter COS301Y/501/0/2009 (the "Study Guide"). Note that $n = 5$.

- Step 0. The grammar is already in the correct form, namely CNF. Put $i = 0$.

- Step 1. Put $i = 1$. We have five substrings of length 1.

Substring	All producing nonterminals
b	B by $B \longrightarrow b$
b	B by $B \longrightarrow b$
a	A by $A \longrightarrow a$
a	A by $A \longrightarrow a$
b	B by $B \longrightarrow b$

- Step 2. $i \neq 5$, thus go to step 1.

- Step 1. Put $i = 2$. We have four substrings of length 2.

Substring	All producing nonterminals
bb	$A \quad (A \longrightarrow BB)$
ba	none (no production with BA on right-hand side)
aa	none (no production with AA on right-hand side)
ab	$B \quad (B \longrightarrow AB)$; $\quad S \quad (S \longrightarrow AB)$

- Step 2. $i \neq 5$, thus go to step 1.

- Step 1. Put $i = 3$. We have three substrings of length 3.

Substring	All producing nonterminals
bba	none — neither $b(ba)$ nor $(bb)a$.
baa	none — neither $b(aa)$ nor $(ba)a$.
aab	$(a)(ab)$: $\quad S \quad (S \longrightarrow AB)$; $\quad B \quad (B \longrightarrow AB)$

- Step 2. $i \neq 5$, thus go to step 1.

- Step 1. Put $i = 4$. We have two substrings of length 4.

Substring	All producing nonterminals
$bbaa$	None. We cannot produce $b(baa)$ or $(bb)(aa)$ or $(bba)a$.
$baab$	$(b)(aab)$: $\quad A \quad (A \longrightarrow BB)$

- Step 2. $i \neq 5$, thus go to step 1.

- Step 1. Put $i = 5$. We have one substring of length 5.

Substring	All producing nonterminals
$bbaab$	$(bb)(aab)$: $\quad S \quad (S \longrightarrow AB \Longrightarrow BBB \Longrightarrow BBAB \Longrightarrow BBAAB)$

- Step 2. We see that $i = 5$ $(=n)$ and that S **is** a producing nonterminal for the word $bbaab$. Thus, the word can be produced by the given grammar.

©
UNISA 2009