Exercises for ACS 1, Fall 2005, sheet 1: Solutions

Exercise 1 (30 points): Give a formal definition of the notion of an "alphabetical enumeration" of Σ^*, which we introduced informally in the lecture. Note: there are many ways to formalize this operation of listing all words from Σ^*.

Solution: Let $\Sigma = \{a_1, ..., a_n\}$, and let $w, v \in \Sigma^*$. We say that w **comes before** v **in the alphabetical enumeration order**, written $w < v$, iff

$|w| < |v|$

or

$|w| = |v| = n > 0$ and $w = ur$ and $v = us$ for some $u \in \Sigma^*$, $r, s \in \Sigma^+$, where the first symbol of r is a_i and the first symbol of s is a_j, and $i < j$.

Then a mapping $\alpha: \Sigma^* \to \mathbb{N}$ is an **alphabetical enumeration** of Σ^* if it is bijective and respects alphabetical enumeration order, that is, for all $w, v \Sigma^*$: $w < v \Leftrightarrow \alpha(w) < \alpha(v)$.

Exercise 2 (30 points): Two sets S and T have (by definition) equal cardinality, written $| S | = | T |$, if there exists a bijective mapping between them. Show that $| [0,1] | = | \mathbb{R} |$, where [0,1] is the set of real numbers between 0 and 1, including 0 and 1.

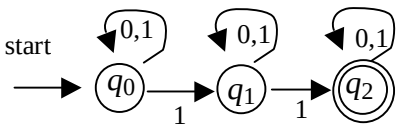
Solution. We first fix a bijective mapping f between (0,1) – the set of reals between 0 and 1, excluding 0 and 1 – and $\mathbb{R}$. There are many such mappings, for instance, $f(x) = \tan(x/\pi + 1/2)$. Let $x_1, x_2, x_3 ... = 1/4, 1/8, 1/16, ...$. Transform f into a bijective mapping $f' : [0,1) \to \mathbb{R}$ (where [0,1) is the set of reals between 0 and 1, including 0 and excluding 1), by putting

$f'(0) = f(x_1)$,
$f'(x_i) = f(x_{i+1})$ for $i \geq 1$,
$f'(x) = f(x)$ for any x not equal to 0 or any of the x_i.

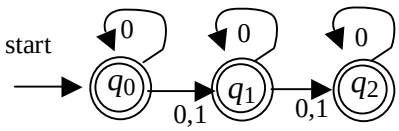
By repeating this trick one obtains a bijective mapping $f'' : [0,1] \to \mathbb{R}$.

Exercise 3. (a) (10 points) Design an NFA for the language $L = \{w \in \{0,1\}^* \mid w$ contains at least 2 ones$\}$. Present your NFA by a graph representation (transition diagram). (b) (10 points) Design an NFA for the language $L = \{w \in \{0,1\}^* \mid w$ contains at most 2 ones$\}$.

Solution: (a) One possibility is

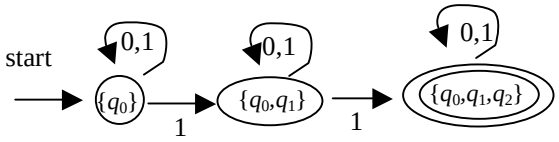


(b)

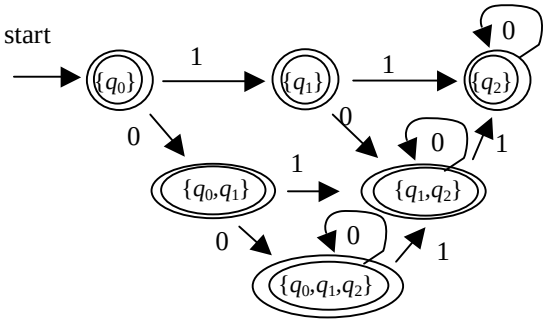


Exercise 4 (20 points). Transform your two NFAs from exercise 3 into equivalent DFAs (with dead state if appropriate) by the subset construction. Present your solution DFAs through their transition graphs.

Solution. (a)



(b)

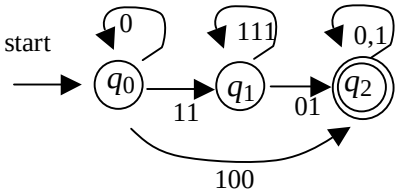


Exercises for ACS 1, Fall 2005, sheet 2

Return solutions in paper form on, in the lecture, on Wednesday, September 28

Note: a maximum of 100 points is accredited for this sheet.

Exercise 1 (50 points). A word-labelled DFA is a generalization of DFAs in whose transition graphs the transition arrows may be labelled by any words from Σ^+. An example might look like this:



(a) (40 points) Give an analog of the definitions 3.1, 3.2 and 3.3 from the lecture notes that capture word-labelled DFAs (there might be several plausible definitions – the purpose of this exercise is not to find the "correct" one, but to train writing formal definitions). Make sure that your definitions define a mechanism that is actually deterministic, that is, on a given input word at most one path through the transition graph is possible. Furthermore, **make sure that your definition** (specifically, for the extended transition function) **is well-defined**. This will require a proof.

(b) (10 points) Prove the the languages accepted by word-labelled DFAs are the regular languages.

Solution. (a)

Def. 1 A **word-labelled DFA** A is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{q_0, q_1, ..., q_n\}$ is a finite set of *states*, Σ is the alphabet of *input symbols*, $\delta: Q \times \Sigma^+ \to Q$ is a partial function, the *transition function*, q_0 is the *start state*, and $F \subseteq Q$ is the set of *accepting states*. It must hold that

$\forall q \in Q\ \forall w, v \in \Sigma^*:\ \delta(q, w)$ and $\delta(q, v)$ are defined $\to v$ is not a proper initial word of w.

Note: this condition assures determinism.

Def. 2 Let $A = (Q, \Sigma, \delta, q_0, F)$ be a word-labelled DFA. Define the **extended transition function** $\hat{\delta}: Q \times \Sigma^* \to Q$ by induction over words through $\forall q \in Q:\ \hat{\delta}(q, \varepsilon) = q$ and $\forall q \in Q\ \forall w \in \Sigma^*\ \forall v \in \Sigma^+:\ \hat{\delta}(q, wv) = \delta(\hat{\delta}(q, w), v)$, if defined. Use notation $\delta(q, w)$ for $\hat{\delta}(q, w)$.

Proposition: Definition 2 is well-defined.

Proof: This follows immediately from the following lemma: If $w \in \Sigma^*$ and there exits n such that $w = v_1 v_2 ... v_n$, all $v_i \in \Sigma^+$, and there exist states $q_0, q_1, ... q_n$ such that all $\delta(q_i, v_{i+1}) = q_{i+1}$ are defined, then the number n and the words $v_1, v_2, ... v_n$ and states $q_0, q_1, ... q_n$ are unique with this property.

Proof of lemma: Assume that for some $w \in \Sigma^*$ and there exits n such that $w = v_1 v_2 ... v_n$, all $v_i \in \Sigma^+$, and there exist states $q_0, q_1, ... q_n$ such that all $\delta(q_i, v_{i+1}) = q_{i+1}$ are defined, and furthermore, there exits m such that $w = v'_1 v'_2 ... v'_m$, all $v'_j \in \Sigma^+$, and there exist states $q_0 = q'_0$, $q'_1, ... q'_m$ such that all $\delta(q'_i, v'_{i+1}) = q'_{i+1}$ are defined. Then $v'_1 = v_1$ or v'_1 is a proper initial word of v_1 or v_1 is a proper initial word of v'_1. The latter two cases are impossible due to Definition 1. Therefore $v'_1 = v_1$. An iterated use of this argument yields the statement.

Def. 3 A word-labelled DFA A **accepts** a word w if $\delta(q_0, w) \in F$. If $\delta(q_0, w) \notin F$, the word is **rejected**. The set of words $L(A) = \{w \in \Sigma^* \mid A \text{ accepts } w\}$ is the **language accepted by** A.

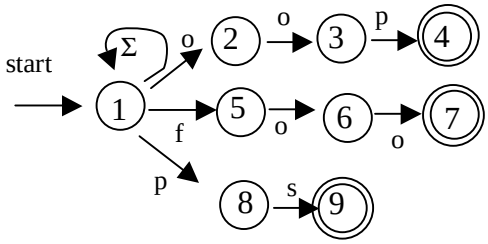
(b) Because the word-labelled DFAs generalize the DFAs, it is clear that every language accepted by a word-labelled DFA is also accepted by a DFA. Conversely, let $A = (Q, \Sigma, \delta, q_0, F)$ be a word-labelled DFA. Construct from A a standard DFA $A' = (Q', \Sigma, \delta', q_0, F)$ as follows.

1. Put $Q^* = Q$, $\delta^* = \{(q, a, p) \mid \delta(q, a) = p, a \in \Sigma\}$
2. For all defined transitions $\delta(p_0, w) = p_n$, $w = a_1 ... a_n$, $n > 1$, $a_i \in \Sigma$, extend Q^* by new states $p_1, ..., p_{n-1}$, and extend δ^* by $\{(p_i, a_i, p_{i+1}) \mid i = 0, ..., n-1\}$.
3. Put $Q' = Q^*$, $\delta' = \delta^*$.

Exercise 2 (20 points). In analogy to the `web – ebay` example from the lecture, design a DFA that finds all occurences of `oop`, `foo`, and `ps` in a text. Present the NFA from which you start your construction by a transition diagram (points) and the resulting DFA by a transition table (points). Finally, give the state DFA sequence that is yielded by input

`soopsssfoos`. Present this state sequence in the format of iterated triples state – (symbol) – nextstate

Solution. We have S = {o, p, s, f}. NFA:



DFA transition table:

	o	p	s	f
→1	12	18	1	15
12	123	18	1	15
123	123	148	1	15
148*	12	18	19	15
15	126	18	1	15
126	1237	18	1	15
1237*	123	148	1	15
18	12	18	19	15
19*	12	18	1	15

Transition sequence on `soopsssfoos`:

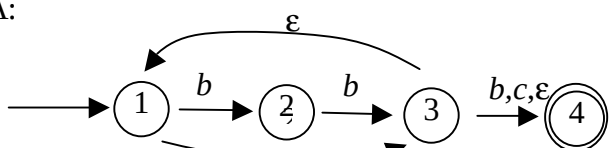
1 (s) 1 (o) 12 (o) 123 (p) 148* (s) 19* (s) 1 (s) 1 (s) 1 (f) 15 (o) 126 (o) 1237* (s) 1

Exercise 3. (20 points) Let Σ be some alphabet and W a finite set of words over Σ. Show that $L = \{w \in \Sigma^* \mid w$ contains no subword $v \in W\}$ is regular.

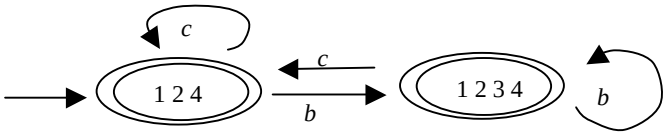
Solution. It is easy to construct an NFA A for L^c, the language of all words over Σ that contain some subword $v \in W$. (This NFA would have the structure of the NFA given in the solution to exercise 6). By the subset construction, one obtains an equivalent DFA A' for L^c. Replace the set of accepting states of A' by its complement to obtain a DFA for L. Thus L is accepted by some DFA and hence is regular.

Exercise 4 (22 points). Design a DFA that accepts the language over S = {b, c} denoted by $((bb)^*(b+c+\varepsilon))^*$, by (a) (10 points) designing first an ε-NFA for this language, using the methods from the proof of proposition 3.4, possibly with simplifications that suggest themselves, then (b) (10 points) deriving an equivalent DFA from that by the subset construction. Represent your automata by transition diagrams. (c) (2 points) Is there a simpler DFA than the one obtained from the subset construction?

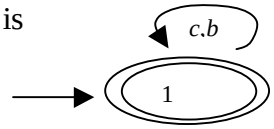
Solution. (a) Merging pure ε-chains, the methods from the proof of proposition 3.4 yield the following ε-NFA:



(b) The subset construction yields



(c) A simpler DFA is



Exercises for ACS 1, Fall 2005, sheet 3: Solution sheet

Exercise 1. Let $\Sigma = \{0,1\}$. Prove or disprove the following two claims (L_i are language variables):

(a) $(L_1 + L_2)^* =_\Sigma (L_1^* L_2^*)^*$

(b) $(L_1 + L_2)^* L_3^* =_\Sigma L_1 L_3^* + L_2 L_3^*$

Solution. (a) Claim is true. Using Corollary 3.7, we have to show that $L((\boldsymbol{a} + \boldsymbol{b})^*) = L((\boldsymbol{a}^* \boldsymbol{b}^*)^*)$ for ordinary regexps $(\boldsymbol{a} + \boldsymbol{b})^*$ and $(\boldsymbol{a}^* \boldsymbol{b}^*)^*$ over $\Sigma = \{a, b\}$. We have to demonstrate two set inclusions:

(i) Let $w \in L((\boldsymbol{a} + \boldsymbol{b})^*)$. Then $w \in \{a, b\}^*$, that is, $w = x_1 \ldots x_n$, $n \geq 0$, $x_i \in \{a, b\}$. Because $a \in L(\boldsymbol{a}^*) = L(\boldsymbol{a}^* \varepsilon) \subseteq L(\boldsymbol{a}^* \boldsymbol{b}^*)$ and $b \in L(\boldsymbol{b}^*) = L(\varepsilon \boldsymbol{b}^*) \subseteq L(\boldsymbol{a}^* \boldsymbol{b}^*)$, we have $x_i \in L(\boldsymbol{a}^* \boldsymbol{b}^*)$ and therefore $w \in L((\boldsymbol{a}^* \boldsymbol{b}^*)^*)$.

(ii) Let $w \in L((\boldsymbol{a}^* \boldsymbol{b}^*)^*)$. Then $w \in \{a, b\}^*$, that is, $w \in L((\boldsymbol{a} + \boldsymbol{b})^*)$.

(b) Claim is not true. Proof by contradiction. If it were true, then for ordinary regexps $(\boldsymbol{a}+ \boldsymbol{b})^* \boldsymbol{c}^*$, $\boldsymbol{a}\, \boldsymbol{c}^* + \boldsymbol{b}\, \boldsymbol{c}^*$ it would hold that $L((\boldsymbol{a}+ \boldsymbol{b})^* \boldsymbol{c}^*) = L(\boldsymbol{a}\, \boldsymbol{c}^* + \boldsymbol{b}\, \boldsymbol{c}^*)$. But clearly $\varepsilon \in L((\boldsymbol{a}+ \boldsymbol{b})^* \boldsymbol{c}^*)$, whereas $\varepsilon \notin L(\boldsymbol{a}\, \boldsymbol{c}^* + \boldsymbol{b}\, \boldsymbol{c}^*)$, because any word in this language must start with a or b. So $L((\boldsymbol{a}+ \boldsymbol{b})^* \boldsymbol{c}^*) \neq L(\boldsymbol{a}\, \boldsymbol{c}^* + \boldsymbol{b}\, \boldsymbol{c}^*)$, and thus the claim is false.

Exercise 2. Prove that the language $L = \{0^n \mid n = pq$ for two primes $p, q\}$ is not regular.

Solution. Assume L is regular. Then by the pumping lemma, there exists a constant c such that, if $w \in L$, $|w| > c$, w can be written as uvx, $r = |v| > 0$, such that $uv^i x \in L$ for all $i \geq 0$. Let p, q such that $n = pq > c$. Then $0^{pq} \in L$ and by the pumping lemma, $0^{pq+ir} \in L$ for all $i \geq 0$. Specifically, for $i = pq$ we obtain $0^{pq(1+r)} \in L$. But $pq(1+r)$ is not the product of two primes,

so by the definition of L, $0^{pq(1+r)} \notin L$. Contradiction, therefore the assumption that L is regular is wrong, therefore L is not regular.

Exercise 3. Prove or disprove the following conjecture:

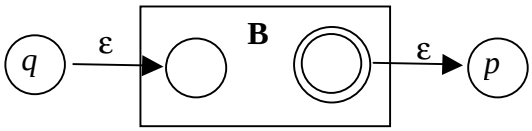
Let M be some regular language over $\Sigma = \{0,1\}$. Define $L_{|M|} = \{0^n \in \{0\}^* \mid n = |v| \text{ for some word } v \in M\}$. Then L is regular.

Solution. The conjecture is true. Consider the (unique!) homomorphism h: $\{0,1\} \to \{0\}$. Then $L_{|M|} = h(M)$ and by Proposition 3.11, $L_{|M|}$ is regular.

Exercise 4. Prove or disprove the following conjecture:

Let M, N be regular languages over $\Sigma = \{0,1\}$. Define $L = \bigcup_{k=|v| \text{ for some } v \in M} N^k$. Then L is regular.

Solution. The conjecture is true. Consider the language $L_{|M|}$ over $\{0\}$ from Exercise 3, which is regular and therefore can be represented by some DFA A whose transitions are labelled by 0. Let B be an ε-NFA for N, which has a single accepting state, no arcs leading into the starting state and no arcs leaving the accepting state (such an ε-NFA exists, see the proof of proposition 3.4 in the lecture notes). In A, insert into every transition $q \xrightarrow{0} p$ a copy of B via two ε-links, as in the figure:



Then make all accepting states from the B copies non-accepting. It is clear that the obtained ε-NFA accepts L.

Exercise 5. Let h be the homomorphism $h(a) = 01$, $h(b) = 0$. Find $h^{-1}(L)$, where $L = (\mathbf{10}+\mathbf{1})^*$.

Solution. First, observe that $\varepsilon \in L$ and thus $\varepsilon \in h^{-1}(L)$ because by definition of a homomorphism, $h(\varepsilon) = \varepsilon$. For non-empty $w \in L$, observe that w must begin with a 1. Assume $w = h(v)$. If the first symbol of v is an a, then w would begin with 01; if the first symbol of v is a b, then w would begin with 0. In neither case does w begin with 1, so the assumption $w = h(v)$ cannot hold, so $h^{-1}(w) = \varnothing$. Therefore, $h^{-1}(L) = \{ \varepsilon \}$.

Exercises for ACS 1, Fall 2005, sheet 4: Solution sheet

Exercise 1 (30 points). Let $\varepsilon \neq w = x_1 \ldots x_n \in \Sigma^*$. Construct the minimal DFA for $L = \{w\}$, by specifying the equivalence classes $[x]_{R_L}$ from the Myhill-Nerode theorem (specify them the

equivalence classes by their extension, that is, for each class describe the set of words that it contains. Prove that your equivalence classes exhaust Σ^* and satisfy the conditions of R_L!). Give a graph representation of your DFA.

Solution. The equivalence classes are (obviously...) $[\varepsilon] = \{\varepsilon\}$, $[x_1] = \{x_1\}$, ..., $[x_1...x_n] = \{x_1...x_n\}$, and (for any $a \in \Sigma$) $[x_1...x_n\, a] = \Sigma^* \setminus ([\varepsilon] \cup [x_1] \cup ... \cup [x_1...x_n])$. This obviously exhausts Σ^*. To show that these sets are actually the R_L – classes, we first show that all members in $[x_1...x_n\, a] = \Sigma^* \setminus ([\varepsilon] \cup [x_1] \cup ... \cup [x_1...x_n])$ are R_L – equivalent. This is clear because $\Sigma^* \setminus ([\varepsilon] \cup [x_1] \cup ... \cup [x_1...x_n])$ contains all words u that are not an initial subword of w, and therefore any extension uv is not in L. By this we have shown that our classes are a refinement of R_L. It remains to show that no two of them are subsets of some R_L – class. To this end, it suffices to show that the representatives we have given are pairwise not R_L – equivalent. This is straightforward: for each pair of the singleton classes, say $[u]$ and $[v]$, where $u \neq v$ are initial subwords of w, consider y where $w = uy$. Then clearly $uy \in L$ but $vy \notin L$, so not $u\, R_L\, v$. Furthermore, $x_1...x_n\, a$ is not R_L – equivalent to any of the initial subwords of w, because each of the latter can be right-extended to become an element of L, whereas $x_1...x_n\, a$ cannot. All in all, we have shown that our sets are indeed the R_L – equivalence classes.

Note: an alternative proof would be to construct the minimal DFA first via the table-filling algorithm, then deduce from it the sets $[x]_{R_L}$.

The minimal DFA is so obvious that I do not draw it here – (but in your solutions you should).

Exercise 2 (40 points). What are the Myhill-Nerode equivalence classes of R_L for $L = \{0^n1^n \mid n \geq 1\}$? (Prove your answer!)

Solution. For some word $u \in \{0,1\}^*$ let $\text{ext}(u) = \{v \in \{0,1\}^* \mid uv \in L\}$ be the *extension set* of of u into words of L. It is clear that $u\, R_L\, v$ iff $\text{ext}(u) = \text{ext}(v)$. Call a word $u \in \{0,1\}^*$ *extendable* if for some $v \in \{0,1\}^*$ it holds that $uv \in L$. It is clear that u is extendable iff $\text{ext}(u) \neq \emptyset$. Furthermore it is clear that u is extendable iff u is of the form 0^a1^b, where $a \geq 1$, $b \geq 0$ and $a \geq b$.

Case 1: u is of the form 0^a, where $a \geq 1$. Then $\text{ext}(u) = \{0^c1^d \mid d = c + a\}$. For different a, these are different extension sets, therefore not $0^a\, R_L\, 0^b$ for $a \neq b$, and these words lie in different equivalence classes.
Case 2: u is of the form 0^a1^b, where $a \geq 1$, $b \geq 1$ and $a \geq b$. Then $\text{ext}(u) = \{1^d \mid d = a - b\}$. We find that for such words, $0^a1^b\ R_L\, 0^{a'}1^{b'}$ iff $a - b = a' - b'$. Furthermore, the extension classes we get for case 2 words are different from extension classes from case 1, therefore the equivalence classes represented by case 2 words are different from case 1 equivalence classes.
Case 3: u is not extendable. Then clearly $u\, R_L\, v$ iff v is not extendable, and $[u]_{R_L}$ is different from any of the case 1 or 2 classes.

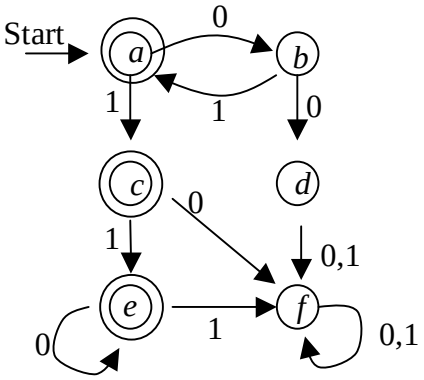
Assembling, we get the following equivalence classes:

1. $[0^a]_{R_L} = \{0^a\}$,

2. $[0^a1^b]_{R_L} = \{0^{a'}1^{b'} \mid a - b = a' - b'\}$,
3. $[1]_{R_L}$ = {0,1}* minus the union of all classes from 1. and 2.

Because we have infinitely many equivalence classes, L is not regular.

Exercise 3 (30 points). Minimize the DFA shown in the figure by using the table filling method. Deliverables: the filling table, the set of states of the minimal DFA, and a graph representation of the minimal DFA.

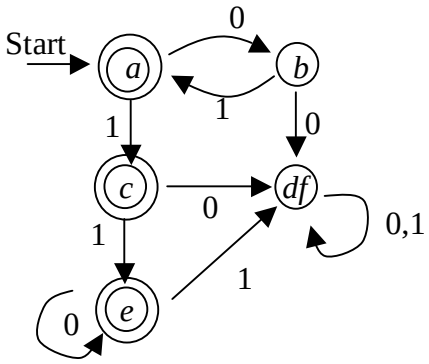


Solution. Manual labour by accurately following the recipe...

Table:

	a	b	c	d	e
b	x_1				
c	x_3	x_1			
d	x_1	x_2	x_1		
e	x_2	x_1	x_2	x_1	
f	x_1	x_2	x_1		x_1

Minimal DFA:



New states: $\{a\}$, $\{b\}$, $\{c\}$, $\{d, f\}$, $\{e\}$

Challenge (no points, for sharpening your claws – really tricky). Show that if L is regular, then also $H = \{u \mid$ there exists a word v, $|u| = |v|$, and $uv \in L\}$ is regular. *Hint*: construct an ε-NFA B for H out of a DFA A for L. Capitalize on the insight that $u \in H$ iff some suitable v with $|u| = |v|$ can be "guessed" (which is something ε-NFA's can do) such that uv leads to an accepting state in A. The difficulty is to ensure that $|u| = |v|$. Think of an ε-NFA that runs two processes in parallel (in a suitable product space of states), one for reading in u, and another one for "guessing" a successful continuation run for v in A.

Exercises for ACS 1, Fall 2005, sheet 5: Solution sheet

Exercise 1 (15 points). Give a CFG for the language of the regular expression (0*10)*.

Solution: A set of production rules that works is the following:

$S \to \varepsilon \mid SS$ comment: this takes care of the outer *
$S \to Z10$
$Z \to \varepsilon \mid ZZ \mid 0$ comment: rules in the last two lines take care of generating the words 0*10

You could also use the generic method that you are asked for in the next exercise, but the resulting grammar would contain more rules and variables.

Exercise 2 (30 points). Describe a generic method by which a CFG for the language $L(E)$ of any regular expression E can be constructed from E. *Hint*: use induction on the structure of E.

Solution. A method to come up with a CFG for any language described by a regular expression E over an alphabet Σ (or T, if we think of it from the angle of CFGs) is to combine CFG's for subexpressions E' of E, by induction over the structure of E:

Basis:
For $E = \emptyset$, ε, or a (where a $\in \Sigma$) it is (almost) trivial to find CFGs $G_\emptyset = (V_\emptyset, \Sigma, P_\emptyset, S_\emptyset)$, $G_\varepsilon = (V_\varepsilon, \Sigma, P_\varepsilon, S_\varepsilon)$, $G_a = (V_a, \Sigma, P_a, S_a)$, that generate the corresponding languages, so I will skip this here.

Induction:
(union) Let E, F be regexps over Σ, with grammars $G_E = (V_E, \Sigma, P_E, S_E)$, $G_F = (V_F, \Sigma, P_F, S_F)$ for the languages of E and F. Without loss of generality, we may assume that V_E and V_E are disjoint. Then we get a obvious grammar for the language of $(E + F)$ by $G_{E+F} = (V_E \cup V_E \cup \{S^*\}, \Sigma, P_E \cup P_F \cup \{ S^* \to S_E , S^* \to S_F\}, S^*)$, that is, we create a new start symbol from which either the old start symbol of the CFG for the language of E or the old start symbol of the CFG for the language of F can be produced.
(concatenation) Similar to union, except the new ruleset is $P_E \cup P_F \cup \{ S^* \to S_E S_F\}$.
(Kleene star) A grammar for the language of E^*, given $G_E = (V_E, \Sigma, P_E, S_E)$, is $G_{E^*} = (V_E, \Sigma, P_E \cup \{ S_E \to S_E S_E , S_E \to \varepsilon \}, S_E)$.

Exercise 3 (20 points). Construct a CFG for the language over $T = \{0,1,2\}$,
$L = \{w2w^R2 \mid w$ is in $(0+1)^+$, w^R is the reverse of $w\}$. (For a regexp E, $L(E^+)$ is $L(E^*) - \{\varepsilon\}$.)

Solution. One possibility is

$S \to 1A12 \mid 0A02$
$A \to 1A1 \mid 0A0 \mid 2$

Exercise 4 (50 points) Let G be a CFG. Show that there exists a constant K (which depends on G), such that, if $w \in L(G)$, $w \neq \varepsilon$, then there exists a derivation with at most $K|w|$ steps.

Solution. Showing the claim is easy if G has no ε-productions (that is, productions of the form $A \to \varepsilon$). Because in this case, consider any parse tree t for w. Its leaves are the symbols from w. Generally, a tree with n leaves cannot have more than $2n$ linear chains (this maximal number is attained when the tree is binary). Consider any linear chain in t. It must consist solely of variable nodes, with possibly the last node being labelled by a terminal if the chain ends in a leaf. Let $k = |V|$ be the number of variables in G. If a chain in t contains more than k

variable nodes, at least one variable A occurs twice and the chain can be shortend by cutting out the segment between the two occurences of A, and joining the two A-nodes, resulting in another parse tree for w. So we can shorten all linear chains to a length bounded by $k + 1$. The total number of nodes of the shortened parse tree is therefore bounded by $2(k + 1)\ |w|$. Putting $K = 2(k + 1)$ therefore gives us the desired bound, because for any parse tree with m nodes we can find a derivation with m-1 steps.

Now assume that G does have ε-productions. Let t be some parse tree for w, with node set N. Consider the subset N' of N that consists of all nodes that have a non-ε-labelled leaf beneath them or are themselves non-ε-leaves. Consider the labelled tree t' that you get when you take away from t all nodes not in N'. This tree has $|w|$ leaves labelled by the symbols from w, from left to right. Notice that if n' is a node within a linear chain in t', then within t this node is either also occuring in a linear chain, or if not, all branches beneath n' in t that are not in t' end in ε leaves. By a similar argument as above (in the case of a grammar with no ε-productions), shortening linear chains in t' at repetitions of variable labels gives a tree t'' whose size is bounded by $K'\ |w|$, where K' is some constant that depends only on $|V|$. Now re-complete t'' by appending, at all nodes n'' that within t had children n not in N', those children n and all their descendents. Obtain t'''. All of these children n are nodes beneath which only ε-leaves can be found. Let b be the maximum length of any production in G. By re-extending t'' into t''', at most $K'\ |w|\ b$ such nodes n were added to t'' (plus the descendants of the nodes n). Consider any such node n in t beneath which only ε-leaves can be found, and all its descendants. This is a subtree t_n of t with only ε-leaves. Any branch beneath n that is longer than $|V|$ must contain repeated variable nodes and can be shortened. So we can condense t_n into a tree t_n' of depth at most $|V|$. If b is the maximum length of any production in G, then the number of nodes of t_n' is bounded by $b^{|V|}$. If we replace all subtrees beneath nodes n in t''' by the condensed trees t_n', we get a tree t'''' that has at most $K'\ |w| + b^{|V|}\ K'\ |w|\ b =: K\ |w|$ nodes. But t'''' is a parse tree for w, and thus we have procured a derivation of w in G with at most $K\ |w|$ steps.

Exercises for ACS 1, Fall 2005, sheet 6: Solution sheet

Exercise 1. Give a formal definition of something like a "context-free graph grammar" (CFGG), where the result of a derivation is not a word, but an undirected graph (whose nodes or links are not labelled). Like in ordinary CFGs, a derivation should start with a start variable S. You are essentially free to invent your personal brand of a CFGG – this exercise aims only at sharpening your skills to compose rigorous definitions, not at finding the "correct" definition. However, to make your task not too simple, your CFGG should be expressive enough to allow grammars that can generate (i) all binary trees, and (ii) all fully connected finite graphs.

Note: Feel invited to become inventive! don't stick too closely to the definitions of ordinary CFGs. Anything goes ... as long as you make it formal and precise, and as long as your grammar can derive the languages (i) and (ii).

Deliverables: **(a, 70 points):** An CFGG definition, including a notion of derivations and of accepted graph languages, analog to Definitions 4.1, 4.2, 4.3 from the lecture notes. **(b, 20 points):** A specific graph grammar whose language is the set of all finite binary trees. Demonstrate your grammar by an illustrative derivation. **(c, 20 points):** A specific graph grammar whose language is the set of all fully connected finite graphs. Demonstrate your grammar by an illustrative derivation.

Use the following definition and terminology for undirected (finite) graphs:

Definition. An undirected graph is a pair $U = (N, E)$, where N is a finite, nonempty set of *nodes* and $E \subseteq N \times N$ is the set of edges. E must be symmetric (that is, $(n, n') \in E \Rightarrow (n', n) \in E$) and antireflexive (that is, $(n, n) \notin E$ for all nodes n). We write $[n, n'] \in E$ as a shorthand for $(n, n') \in E \wedge (n', n) \in E$. An undirected graph is *connected* if for any pair (n, n') of nodes, there exists a path in the graph that leads from n to n'. An undirected, connected graph is a binary tree if either $U = (\{n_0\}, \varnothing)$ or, if $| N | > 1$, there exists exactly one $n_{\text{root}} \in N$ that has exactly two neighbors, and any other node has either exactly one neighbor (then it is a *leaf node*) or three neighbors (then it is an *internal node*). An undirected graph is *fully connected* if for any pair (n, n') of nodes, it holds that $[n, n'] \in E$.

If (N, E) is a graph, A and B are sets, then any mapping $\nu: N \to A$ is called a *node labelling (with label set A).* For a graph with a node labelling ν we write (N, E, ν).

Solution. **(a)** One possibility would be the following:

Definition. (CFGG) A *context-free graph grammar* (CFGG) is a quadruple $G = (V, T, P, S)$, where

1. V is a finite set of *variables* (also called *nonterminals*),
2. $T \notin V$ is the *terminal symbol*,
3. $S \in V$ is a *start symbol*, and
4. P is a finite set of *productions* (also *rules*), each of the form $A \to \alpha$, where A is a variable and $\alpha = (N, E, \nu, \pi)$ consists of the following parts:
    1. an undirected, node-labelled graph (N, E, ν), with labels from $V \cup \{T\}$, called the *heart* of the rule,
    2. for each node $n \in N$ of the heart a finite (possibly empty) set $\pi(n)$ of *graph matching patterns*. A graph matching pattern is itself an undirected graph (N', E'), where $|N'| \geq 2$, where there is a specifically marked *reference node* $r \in N'$, and a *match node* $m \in N'$, with $r \neq m$. Thus, $\pi(n)$ consists of a finite set of structures of the form $((N', E'), r, m)$.

Definition. (derivations). Let $G = (V, T, P, S)$ be a CFGG. Let (N, E, ν), (N', E', ν') be two node-labelled graphs, with label set $V \cup \{T\}$ each. Then (N', E', ν') is derived from (N, E, ν) via G, written $(N, E, \nu) \Rightarrow_G (N', E', \nu')$, if there exists a rule $A \to \alpha$ in P, such that

1. there exists a node $n \in N$, $\nu(n) = A$,
2. (N', E', ν') can be obtained from (N, E, ν) through the following steps:
    a. delete from (N, E, ν) the node n (including its connecting links and label information), to obtain (N^*, E^*, ν^*),
    b. add to (N^*, E^*, ν^*) the heart (N_α, E_α) of α (making N_α is disjoint from N^*), obtaining $(N^{**}, E^{**}, \nu^{**}) = (N^* \cup N_\alpha, E^* \cup E_\alpha, \nu^* \cup \nu_\alpha)$,
    c. for each node n_α in N_α, and for each $((N_\pi, E_\pi), r, m) \in \pi(n_\alpha)$, and for each $n^* \in N^*$, add $[n_\alpha, n^*]$ to E^{**} if $((N_\pi, E_\pi), r, m)$ matches r with n and m with n_α, that is, if there exists an injective map $\iota: N_\pi \to N$ with $\iota(r) = n$ and $\iota(m) = n^*$

which is a subgraph map, that is, connections in (N_π, E_π) are mapped on existing connections in (N, E).

Let $\Rightarrow^*_G$ denote the transitive closure of $\Rightarrow_G$.

Definition (language of a CFGG). Let $G = (V, T, P, S)$ be a CFGG. Then an undirected graph (N, E) is in the language of G if there exists a derivation $(\{S\}, \varnothing) \Rightarrow^*_G (N, E, \nu)$, where ν labels all nodes from N by the terminal symbol T.

(b). Put $V = \{S, F\}$ and for the productions set put

1. $S \to (\{n\}, \varnothing, \nu, \pi)$, with $\nu(n) = T$ and $\pi(n) = \varnothing$. [This rule generates the trivial single-node tree]
2. $S \to (\{a, b, c\}, \{[a, b], [a, c]\}, \nu, \pi)$, where $\nu(a) = T$ and $\nu(b) = \nu(c) = F$, and $\pi(a) = \pi(b) = \pi(c) = \varnothing$. [This rule generates a "seed" tree of one root and two descendants].
3. $F \to (\{a, b, c\}, \{[a, b], [a, c]\}, \nu, \pi)$, where $\nu(a) = T$ and $\nu(b) = \nu(c) = F$, and $\pi(a) = (\{r, m\}, \{[r, m]\}, r, m)$, $\pi(b) = \pi(c) = \varnothing$. [This rule extends a "frontier" node a labelled by F by appending two siblings. a is relabelled by T, the siblings become new "frontier" nodes by labelling them with F.
4. $F \to (\{a\}, \varnothing, \nu, \pi)$, where $\nu(a) = T$ and $\pi(a) = (\{r, m\}, \{[r, m]\}, r, m)$. [This rule changes the status of a frontier node into a terminal node by relabelling to T.]

(c). Put $V = \{S, A\}$ and for the production set put

1. $S \to (\{n\}, \varnothing, \nu, \pi)$, with $\nu(n) = A$ and $\pi(n) = \varnothing$. [This rule generates the trivial single-node tree]
2. $A \to (\{a, b\}, \{[a, b]\}, \nu, \pi)$, where $\nu(a) = \nu(b) = A$, and $\pi(a) = \pi(b) = (\{r, m\}, \{[r, m]\}, r, m)$. [This rule adds a new node and connects it to all nodes in the old graph, assuming that the node that is replaced using this rule is already connected to all nodes in the old graph.]
3. $A \to (\{a\}, \varnothing, \nu, \pi)$, where $\nu(a) = T$ and $\pi(a) = (\{r, m\}, \{[r, m]\}, r, m)$. [This rule changes the status of an "active" node into a terminal node by relabelling to T.]

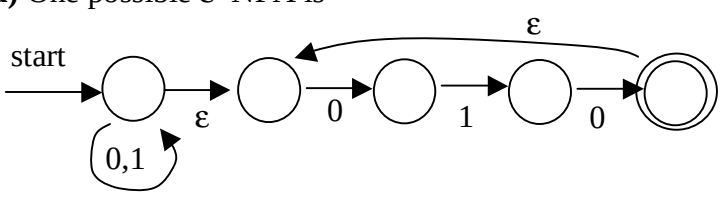
(for **(b)** and **(c)**, you should also furnish an illustrative derivation – omitted here).

Midterm Advanced CS 1, Fall 2005: Solutions

Group A

Problem 1.
(a, 10 points) Construct an ε-NFA for the language
$L = \{w \in \{0,1\}^* \mid w = uv, u \in \{0,1\}^*, v$ is a string of n repeated substrings 010, where $n \geq 1\}$.
Present your automaton by a transition graph.
(b, 10 points) Represent L by a regular expression. Adhere strictly to the syntax of regexps as introduced in the lecture (no parenthesis saving conventions)
(c, 10 points) Give a CFG for L.

Solution. (a) One possible ε–NFA is



(b) $((\mathbf{0} + \mathbf{1})^*)(((\mathbf{01})\mathbf{0})^*))$
(c) One possible grammar: $S \to UV$, $U \to \varepsilon \mid U1 \mid U0$, $V \to W010$, $W \to \varepsilon \mid V$

Problem 2 (15 points): Prove or disprove the following conjecture:

Let Σ be some alphabet. If for a language $L \subseteq \Sigma^*$, all k-initial languages are regular ($k \geq 1$), then L is regular. (For a language L, the k-initial language of L is the set $\{v \in \Sigma^* \mid |v| = k$ and there exists a word $vu \in L\}$.)

Solution. Conjecture is very wrong. By definition of a k-initial language, a k-initial language is finite and therefore regular. So for *any* language L all k-initial languages are regular – regardless of whether L is regular or not.

Problem 3 (20 points): Show that the language $L = \{0^k10^l \mid k \geq 0, l \geq k\}$ is not regular.

Solution: a clear job for the pumping lemma, what else... Assume L is regular. Let n be a pumping lemma constant. Consider the word $w = 0^n10^n$. By the PL, it can be written as $w = xyz$, where $|xy| \leq n$, (which implies y consists solely of 0's and falls into the first 0^n of w) and $|y| \geq 1$. By "pumping" y once and using the statement of the PL, we obtain $w' = 0^{n+|y|}10^n \in L$, which is a contradiction to the definition of L. Therefore the assumption that L is regular must be false, that is, L is not regular.

Problem 4 (20 points) Prove that if L (over Σ) is regular, then the prefix language of L, $\mathrm{Pre}(L) = \{w \in \Sigma^* \mid wu \in L$ for some $u \in \Sigma^*\}$ is also regular.

Solution: Let A be some DFA for L. First determine all accessible states q in A from which there exists some path leading to an accepting state, or which are themselves accepting. Add a new state p to the state set of A, and connect all states q to p by an ε-transition. Make p accepting and make all original states of A non-accepting. The automaton thus obtained clearly accepts $\mathrm{Pre}(L)$.

Note: the suffix language from group B can be dealt with in a similar way, by inserting ε-transitions into A from the start state into all states q.

Problem 5 (30 points) Disprove the following variant of the pumping lemma:

L is a regular language iff there exists a constant n, such that $\forall w \in L$, $|w| \geq n$, we can find a partition $w = xyz$, such that (1) $y \neq \varepsilon$, (2) $|xy| \leq n$, and (3) $\forall k \geq 1$, $xy^kz \in L$.

Note: this claim differs from the PL in two respects. First, it has an "iff" where the PL only has a "$\Rightarrow$", second, condition (3) is stated for $\forall k \geq 1$, whereas the PL as $\forall k \geq 0$.

Solution: We construct a non-regular L for which the rhs. of the iff claim holds. Let M be some non-regular language over Σ_M (which must be infinite because all finite languages are

regular), where Σ_M. does not contain 0. Let N be some infinite regular language over $\Sigma_N = \{0\}$ with $\varepsilon \notin L$ and with PL constant n. and let $N_{>n}$ be the subset of N of all words longer than n (this is also regular because $N_{>n} = N \cap \Sigma_N^0 \cap \Sigma_N^1 \cap ... \cap \Sigma_N^n$). Consider the language $L = N_{>n} M$ over $\Sigma_M \cup \Sigma_N$. Then by construction, the rhs. of the iff statement holds for L (use same n and But L is not regular, which can be seen by considering the homomorphism h: $\Sigma_M \cup \Sigma_N \rightarrow \Sigma_M$, $h(0) = \varepsilon$, $h(a) = a$ for all $a \in \Sigma_M$. It holds that $h(L) = M$. If L were regular, then by closure under homomorphisms M would also be regular, contradiction.

Final Advanced CS 1, Fall 2005

Group A

A maximum of 100 points is accredited.

Problem 1.
(a, 5 points) Describe the language accepted by the ε-NFA *A* from Fig. 1 in a "mathematical set description" of the kind $L = \{w \in \{0,1,2,3\}^* \mid$ ... < your specification goes here > }.
(b, 5 points) Represent $L(A)$ by a regular expression. Adhere to the syntax of regexps as introduced in the lecture (you may use parenthesis saving conventions)
(c, 5 points) Give a CFG for *L*.

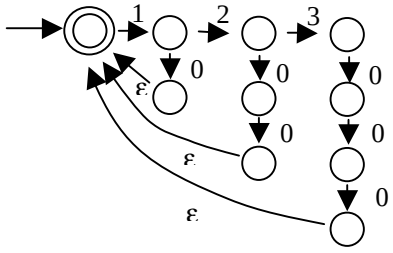
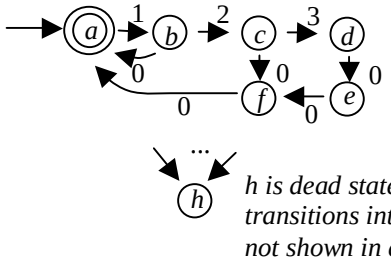


Fig. 1: the ε-NFA *A* for problems 1 and 2

Solution. (a) $L = \{w \in \{0,1,2,3\}^* \mid w$ is a (possibly empty) concatenation of subwords 10, 1200, 123000}
(b) (01 + 1200 + 123000)*
(c) $S \rightarrow \varepsilon \mid SS \mid 01 \mid 1200 \mid 123000$

Problem 2 (25 points). How many states does a minimal DFA for the language accepted by the ε-NFA *A* from Figure 1 have? Prove your answer.

Solution. The following DFA clearly accepts *L*:
We use the table-filling algorithm to show that it is actually minimal. Here is the table that we get:



h is dead state, transitions into it not shown in detail

b	x_1					
c	x_1	x_2				
d	x_1	x_2	x_3			
e	x_1	x_2	x_5	x_3		
f	x_1	x_4	x_2	x_2	x_2	
h	x_1	x_2	x_3	x_4	x_3	x_2
	a	*b*	*c*	*d*	*e*	*f*

The table shows that all states are pairwise distinguishable and therefore the DFA is minimal; that is, the minimal number of states is 7.

Problem 3. (a, 15 points) Under the homomorphism operation, each symbol a from a word w over Σ_1 is replaced by some word $h(a)$ over Σ_2. This is a deterministic operation: $h(w)$ is uniquely determined. Your task: define a non-deterministic version of homomorphisms, where each symbol a from a word w over Σ_1 is replaced by any word from a finite set of words over Σ_2 that depends on a. Re-formulate the original definition 3.13. (1.) of a homomorphism for the non-deterministic case. **(b, 20 points)** Show that the context-free languages are closed under non-deterministic homomorphisms.

Solution. (a) Let Σ_1, Σ_2 be two alphabets. A *non-deterministic homomorphism h* is any function h: $\Sigma_1 \to \mathrm{Pot}_0(\Sigma_2^*)$, where $\mathrm{Pot}_0(X)$ is the set of finite subsets of X. Such h induces another function (also called h) from Σ_1^* to Σ_2^*, by putting $h(\varepsilon) = \varepsilon$ and $h(x_1...x_n) = \{w_1... w_n \mid w_i \in h(x_i)\}$. For $L \subseteq \Sigma_1^*$, $h(L) = \bigcup \{h(w) \in \Sigma_2^* \mid w \in L\}$.

(b) Let $G = (V, \Sigma_1, P, S)$ be a grammar for $L \subseteq \Sigma_1^*$ in CNF. Define a grammar $G' = (V', \Sigma_1, P', S')$ for L' as follows:

1. $V' = V \cup \Sigma$. 2. $S' = S$.
3. Let P' consist of all rules from P (with the rules of the form $A \to a$ now interpreted in G' as variable-variable replacements), plus for each $a \in \Sigma_1$, $w \in h(a)$ a rule $a \to w$ (now interpreted as a variable-terminal_word replacement).

It is clear that G' is a grammar for $L' = h(L)$.

Problem 4 (20 points). A language L over Σ is called *recursively enumerable,* if $L = \bigcup_{n>0} A(n)$, where $A(n)$ is a finite (possibly empty) set of words from Σ^*, and there exists an algorithm that computes the function A: $\mathbb{N} \to \mathrm{Pot}(\Sigma^*)$. That is, on input n, the algorithm prints out the finite number of words contained in $A(n)$. Show that every context-free language L is recursively enumerable, by describing such an algorithm in words. Explain why $L = \bigcup_{n>0} A(n)$ when your algorithm is used.

Solution. There are many ways to achieve such an algorithm. Somehow the algorithm must exhaustively construct all possible derivations (or parse trees). Here, for concreteness, I sketch one possible realization. Let G be a grammar for a CFL L, and let G have k rules. On input n, the algorithm constructs all possible derivations of length less or equal than n. For each of these derivations which ends in an all-terminal sequential form, that is, produces a word $w \in L$, the word w is added to the output $A(n)$. It is clear that $\bigcup_{n>0} A(n) \subseteq L$. Conversely, if $w \in L$, the word has a derivation of some length n, and therefore $w \in A(n)$, thus $w \in \bigcup_{n>0} A(n)$, thus $L \subseteq \bigcup_{n>0} A(n)$.

Problem 5. Formulate the following natural language statements in PL1, using the syntax from Michael Kohlhase's slides. Freely use parentheses if it makes things clear.

(a, 9 points) *Every dog has exactly one mother.* (Use signature $\Sigma^p_1 = \{\text{Dog}\}$, $\Sigma^p_2 = \{\text{Mother}, =\}$, where "=" is intended to denote identity of two individuals; you may use infix notation if you wish)

(b, 6 points) *Two sets are equal iff they contain the same elements.* (Use signature $\Sigma^p_1 = \{\text{Set}\}$, $\Sigma^p_2 = \{\in, =\}$; you may use infix notation if you wish)

Solution. (a) $\forall X\,.\,(\text{Dog}\,X \Rightarrow \exists Y.\,(\text{Mother}(Y, X) \wedge \forall Z\,.\,(\text{Mother}(Z, X) \Rightarrow Z = Y)))$
(b) $\forall X\,.\,\forall Y\,.\,(\text{Set}(X) \wedge \text{Set}(Y) \Rightarrow (\,X = Y \Leftrightarrow \forall Z\,.\,(Z \in X \Leftrightarrow Z \in Y))\,)$

Problem 6 (10 points). For the PL1 formula **A**: $\forall X\,.\,(p(X) \Rightarrow \exists Y.\,q(Y, X))$, where p is a unary and q a binary predicate symbol, design two models $\mathcal{M}_i = <\mathcal{D}_\iota, \mathcal{I}_i>$ (where $i = 1, 2$), each over $\mathcal{D}_\iota = \{1, 2, 3\}$, such that $\mathcal{M}_1 \models \mathbf{A}$ and not $\mathcal{M}_2 \models \mathbf{A}$. Explain in words in each case why your model satisfied the requirements $\mathcal{M}_1 \models \mathbf{A}$ and not $\mathcal{M}_2 \models \mathbf{A}$.

Solution. We have to define two interpretations $\mathcal{I}_i$, one which makes **A** hold and the other not. For $i = 1$, put (for example) $\mathcal{I}_1(p) = \varnothing$, and $\mathcal{I}_1(q)$ anything. Then $<\mathcal{D}_\iota, \mathcal{I}_1> \models \mathbf{A}$, because the premise of the implication in **A** is then void and therefore the implication trivially true. For $i = 2$, put (for example) $\mathcal{I}_2(p) = \mathcal{D}_\iota$, and $\mathcal{I}_2(q) = \varnothing$. Then not $<\mathcal{D}_\iota, \mathcal{I}_2> \models \mathbf{A}$, because the premise of the implication holds for every element of $\mathcal{D}_\iota$, but the consequence for none; the implication is therefore not true for any element of the carrier.

Problem 7 (10 points). Using the rules from the sequent version of natural deduction (Michael Kohlhase's slide 51) derive $\Gamma\ \mathbf{A} \wedge \neg\mathbf{A} \vdash \mathbf{B}$ (for any Γ, **A**, **B**).

Solution. One possible derivation is:

(1) $\Gamma\ \mathbf{A} \wedge \neg\mathbf{A} \vdash \mathbf{A} \wedge \neg\mathbf{A}$	*Ax*
(2) $\Gamma\ \mathbf{A} \wedge \neg\mathbf{A} \vdash \mathbf{A}$	$\wedge\, E_l$ on (1)
(3) $\Gamma\ \mathbf{A} \wedge \neg\mathbf{A} \vdash \neg\mathbf{A}$	$\wedge\, E_r$ on (1)
(4) $\Gamma\ \mathbf{A} \wedge \neg\mathbf{A} \vdash \mathbf{F}_o$	$\mathbf{F}_o\, I$ on (2), (3)
(5) $\Gamma\ \mathbf{A} \wedge \neg\mathbf{A} \vdash \mathbf{B}$	$\mathbf{F}_o\, E$ on (4)