CR15 Final Exam

Notes are allowed, but internet forbidden. Answers must be justified.

Exercise 1: Automata and graphs
Let $G = (V, E)$, with $E \subseteq V \times V$, be a finite directed graph and fix two vertices v_A and v_B in V. An *exponential path* in G is a path from v_A to v_B of length 2^k for some $k \in \mathbb{N}$. Consider the language $L \subseteq V^*$ of *partial exponential paths*, defined as follows: a word $w = v_0 v_1 \dots v_n$ is in L if and only if there exists words $w_0, \dots w_{n+1} \in V^*$ such that the word $w' = w_0.v_0.w_1.v_1 \dots w_n.v_n.w_{n+1}$ is an exponential path.

Question: is L a regular language?

Answer: yes.

Proof: Let $P \subseteq V^*$ be the set of paths from v_A to v_B of length 2^k, for some $k \in \mathbb{N}$. In general P is not regular. However $L = (M\downarrow)$, where $M\downarrow$ is the down–closure of P for the embedding order, and this is regular, as seen in course (by Higman Lemma).

Exercise 2: Games of infinite duration
Consider the following two-player game G, played in alternation by Player L (for Letters) and Player N (for Numbers).

At each turn of the game, Player L chooses a letter in $\{A, B, C, D\}$, then Player N chooses a number in $\{1, 2, 3, 4\}$. The game goes on for ω steps, producing an infinite word $w \in ((A + B + C + D) \cdot (1 + 2 + 3 + 4))^\omega$.

Player N wins the play w if and only if the number k of letters that Player P has chosen infinitely many times in w is equal to the greatest number $n \in \{1, 2, 3, 4\}$ that Player N has played infinitely many times.

1. Show that the game G is determined.
2. Describe an algorithm computing the winning player (no need to run the algorithm explicitely).
3. *(hard)* Describe a winning strategy for the winning player.

Answers:

1. The winning condition is ω-regular, so the game is determined. To prove that the winning condition is ω-regular, we can for instance consider the Müller automaton on alphabet $\Sigma = \{A, B, C, D\} \times \{1, 2, 3, 4\}$, with states $Q = \Sigma$, and transitions $p \xrightarrow{q} q$ for all $p, q \in \Sigma$. Choose any state as initial, and accepting condition be the set of sets of states describing the winning condition, i.e. $F \subseteq Q$ is accepting if and only if the maximal number occuring in F is equal to the number of letters occuring in F.

2. Compute a deterministic parity automaton $\mathcal{A}$ on Σ accepting the winning condition. Then, as seen in course, solve the parity game played on $\mathcal{A}$ where Player L plays letters and Player N plays numbers, and the state is updated deterministically. This yields a finite-memory strategy for the winning player.

3. The winning player is Player N, with the following strategy. The memory structure consists of all possible permutations of $\{A, B, C, D\}$, for instance a possible memory state is $BADC$. When a letter is played by the opponent, the Player N outputs the position of this letter in the permutation, and moves the letter to the first position. For instance from $BADC$, if the letter played is D, then we output 3 and update the memory state to $DBAC$. This is known as the Last Appearance Record, or LAR. It is now straightforward to show that if a number i is the maximal one played infinitely often, then after some finite prefix, the i first letters of the memory state will be played infinitely often, but the $4-i$ last letters will never be played anymore.

Exercise 3: GFG versus unambiguity

A nondeterministic parity automaton $\mathcal{A}$ on alphabet Σ is called *unambiguous* if for any word $w \in \Sigma^\omega$, $\mathcal{A}$ has at most one accepting run on w. It is called *ambiguous* if it is not unambiguous, i.e. if there exists $w \in \Sigma^\omega$ such that $\mathcal{A}$ can accept w with at least two different runs. We consider here that states (and not transitions) are labelled by parity ranks, and a run is simply an infinite sequence of states. Recall that a parity automaton $\mathcal{A}$ with states Q is GFG (or HD) if there is a GFG strategy $\Sigma^* \to Q$ resolving the non-determinism, such that for any word $w \in L(\mathcal{A})$, the run induced by σ on w is accepting.

1. Give two examples of GFG parity automata: one unambiguous and one ambiguous.

2. Give two examples of non-GFG parity automata: one unambiguous and one ambiguous.

A transition $p \xrightarrow{a} q$ is called *residual* if $L(q) = a^{-1}L(p)$, meaning that whenever an infinite word aw is accepted from state p, then w is accepted from state q. If $\mathcal{A}$ is a GFG automaton with GFG strategy σ, we call $\mathcal{A}_\sigma$ the restriction of $\mathcal{A}$ to transitions useful in σ, i.e. transitions of the form $\sigma(u) \xrightarrow{a} \sigma(ua)$, for some word u and letter a. We will also consider that $\mathcal{A}_\sigma$ has the *unique sink property*, i.e. there is at most one state accepting no infinite word (i.e. if there are several such states in $\mathcal{A}_\sigma$ we merge them)..

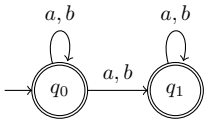
3. Prove that if $\mathcal{A}$ is GFG with strategy σ, then $\mathcal{A}_\sigma$ contains only residual transitions.

4. Prove that if $\mathcal{A}$ is an unambiguous GFG parity automaton, then $\mathcal{A}_\sigma$ is deterministic.

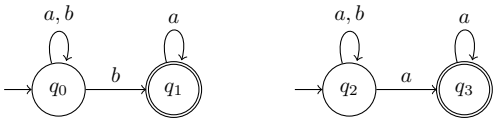
Answers:

All examples will be Büchi automata, with accepting states doubly circled.

1. Any deterministic automaton is GFG and unambiguous.

   Here is an example of a GFG ambiguous automaton: 

2. Here are examples of non-GFG unambiguous and ambiguous automata, respectively.

   Indeed, the first automaton accepts $(a+b)^*ba^\omega$, and must change state exactly on the last b, so only one run is possible, but no GFG strategy exists. The second automaton accepts $(a+b)^*a^\omega$. To change state, it must guess that there is no more b, so it is not GFG. But it can stay longer in the first state, so each word of the language is accepted by infinitely many different runs.

3. First, $\mathcal{A}_\sigma$ is GFG and accepts the same language as $\mathcal{A}$, since σ can still witness the acceptance of any word from $L(\mathcal{A})$. Let $(u, a) \in \Sigma^* \times \Sigma$. Let $p = \sigma(u)$ and $q = \sigma(ua)$. We show that the transition (p, a, q) is residual. First, since $(p, a, q) \in \Delta$, we have $L(q) \subseteq a^{-1}L(p)$. Now, assume that there is $v \in a^{-1}L(p) \setminus L(q)$. We have $av \in L(q)$, so $uav \in L$, but since v cannot be accepted from $\sigma(ua)$, the run given by σ on uav is not accepting. This contradicts the assumption on σ, so we can conclude $L(q) = a^{-1}L(p)$.

4. Assume there is some non-determinism in $\mathcal{A}_\sigma$. This means that there are three states p, q, r with $q \neq r$ and a letter a such that $p \xrightarrow{a} q$ and $p \xrightarrow{a} r$ are transitions in $\mathcal{A}_\sigma$. So there are two words $u, v \in \Sigma^*$ such that $\sigma(u) = \sigma(v = p)$, but $\sigma(ua) = q \neq r = \sigma(va)$. Since $\mathcal{A}_\sigma$ is residual, $L(q) = L(r) = a^{-1}L(p)$. Moreover, since $\mathcal{A}_\sigma$ has the unique sink property, there is an infinite word $w \in L(q) = L(r)$. This means that there are at least two different accepting runs on uaw: one in q after reading ua, and one in r after reading ua. Therefore the automaton is not unambiguous, contradiction. Thus $\mathcal{A}_\sigma$ is deterministic.

Exercise 4: ω-regular languages

Let $\Sigma = \{a, b, c\}$ and $L \subseteq \Sigma^\omega$ be defined as the set of words containing infinitely many factors ab, or finitely many c. For instance $a^\omega \in L$, $(ab)^\omega \in L$, $(abc)^\omega \in L$, but $(ac)^\omega \notin L$.

1. Give an ω-regular expression for L.
2. Give a FO formula for L ($x \leq y$ and $x+1$ are both allowed in the syntax).
3. Give a non-deterministic Büchi automaton for L.
4. Give a deterministic Müller automaton for L.

Let $\Sigma_2 = \{a, b\}$. We define a Wilke algebra $W = (S_{fin}, S_\omega, \cdot, \odot, \omega)$ where $\cdot, \odot, \omega$ are respectively the binary product of S_{fin}, the mixed product and the ω-power in the following way:

$$S_{fin} = \{x, y, z\} \qquad S_\omega = \{0, \alpha, \beta\}$$

$\cdot$	x	y	z
x	y	x	x
y	x	y	y
z	x	y	z

$\odot$	0	α	β
x	0	β	α
y	0	α	β
z	0	α	β

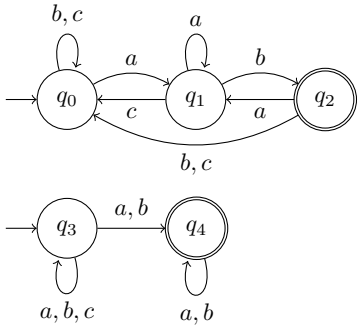
ω	
x	0
y	0
z	α

Furthermore, we define $h : \Sigma_2 \to S_{fin}$ by $h(a) = x$ and $h(b) = z$, and $P \subseteq S_\omega$ by $P = \{\alpha\}$.

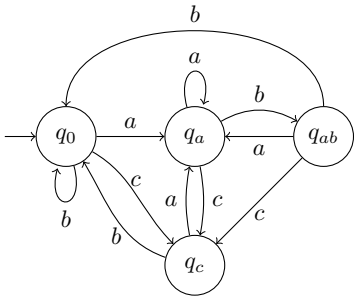
5. Give an ω-regular expression for the language recognized by (W, h, P).

Answers:

1. $(\Sigma^*(ab))^\omega + \Sigma^*(a+b)^\omega$.
2. Let $\varphi_1 = \forall x, \exists y, y \geq x \wedge a(y) \wedge b(y+1)$, describing $(\Sigma^*(ab))^\omega$. Let $\varphi_2 = \exists x, \forall y, y \geq x \Rightarrow \neg c(y)$, describing $\Sigma^*(a+b)^\omega$. We finally set $\varphi = \varphi_1 \vee \varphi_2$.
3. We give a nondeterministic automaton with two initial states. The first component is deterministic, and state q_2 is reached every time ab occurs. The second component guesses a point where no more c occurs, and goes to q_4 to accept. The union of the components accepts the union of their languages.



4. We design an automaton reaching a particular state when ab is read, and another when c is read. It suffices to modify slightly the first component above:



We then define the set $\mathcal{F}$ of accepting sets of states by $F \in \mathcal{F}$ if $q_{ab} \in F$ or if $q_c \notin F$.

5. We will also note h for the unique extensions $\Sigma^* \to S_{fin}$ and $\Sigma^\omega \to S_\omega$. Notice that $h^{-1}(z) = b^*$, and for a finite word u with at least one a, we have $h(u) = x$ if $|u|_a$ is odd, $h(u) = y$ if $|u|_a$ is even. Let L be the language recognized by (W, h, P). Let w be a word with infinitely many a. Then, $w = u_1u_2u_3\ldots$, with $|u_i|_a = 1$ for all i. Therefore, $h(w) = x^\omega = 0$, and $w \notin L$. Now, if $w = ub^\omega$, then $h(w) = h(u) \odot \alpha$. If $|w|_a$ is even, then $h(w) = y \odot \alpha = \alpha \in P$, so $w \in L$. If $|w|_a$ is odd, then $h(w) = x \odot \alpha = \beta \notin P$, so $w \notin L$. To sum up, L is the language of words having a finite and even number of a, i.e. $L = (b^*ab^*a)^*b^\omega$.

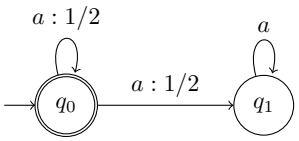
Exercise 5: Probabilistic automata

If $\mathcal{A}$ is a probabilistic automaton and $\lambda \in [0, 1]$ is a real number, we define $L(\mathcal{A}, \lambda) = \{w \in \Sigma^* \mid \mathcal{A} \text{ accepts } w \text{ with probability} \geq \lambda\}$.

1. Give an example of $\mathcal{A}$ and λ such that $L(\mathcal{A}, \lambda)$ is regular but any DFA for it has strictly more states than $\mathcal{A}$.
2. Give $\mathcal{A}$ such that the set $\{L(\mathcal{A}, \lambda) \mid \lambda \in [0, 1]\}$ is infinite but countable.

Answer:

Let $\mathcal{A}$ be the following probabilistic automaton, on alphabet $\Sigma = \{a\}$.



The probability of a^n being accepted is 2^{-n}. Therefore, $L(a, 2^{-10}) = \{a^n \mid n \leq 10\}$, that requires a DFA with 10 states.

Moreover, if $\lambda \in]0, 1]$, then $L(\mathcal{A}, \lambda) = \{a^n \mid n \leq -\log(\lambda)\}$. Therefore, all languages $L(\mathcal{A}, \lambda)$ with $\lambda > 0$ are of the form $\{a^n \mid n \leq k\}$, and they form a countable set.