Technische Universität München **Winter term 2009/10**
I7
Prof. J. Esparza / J. Křetínský / M. Luttenberger **11. Februar 2010**

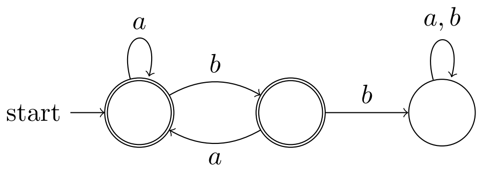
Solution

Automata and Formal Languages – Homework 2

Due 5.11.2009.

Exercise 2.1

Let $\mathcal{A}$ be the following finite automaton:



(a) Transform the automaton $\mathcal{A}$ into an equivalent regular expression, then transform this expression into an NFA (with ε-transitions), remove the ε-transitions, determinize the automaton and then minimize it using the algorithms from the lecture. Check that the resulting automaton is isomorphic to the original one.

(b) Use JFLAP (`http://www.jflap.org/`) to perform the same transformations. Is there any difference?

(c) Let us denote $\Sigma^* = \{a, b\}$. Then $\mathcal{L}(\mathcal{A}) = \Sigma^* \setminus \Sigma^* bb \Sigma^*$. Prove that $\mathcal{A}$ is minimal by

- proving that no equivalence relation $\sim$ on Σ^* of index at most 2 can satisfy $x \sim y \Rightarrow \forall w(xw \in \mathcal{L}(\mathcal{A}) \Leftrightarrow yw \in \mathcal{L}(\mathcal{A}))$ (and thus concluding by Nerode theorem);
- computing the index of $\sim_{\mathcal{L}(\mathcal{A})}$ (and if it is 3 then concluding by Myhill-Nerode theorem);
- performing the *Bisim* algorithm for minimization (and checking the result is isomorphic to the original automaton).

Exercise 2.2

For $n \in \mathbb{N}_0$ let $\text{msbf}(n)$ be the language of all words over $\{0, 1\}$ which represent n w.r.t. to the *most significant bit first* representation where an arbitrary number of leading zeros is allowed. For example:

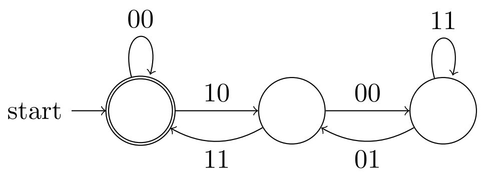
$$\text{msbf}(3) = \mathcal{L}(0^*11) \text{ and } \text{msbf}(0) = \mathcal{L}(0^*).$$

Similarly, let $\text{lsbf}(n)$ denote the language of all *least significant bit first* representations of n with an arbitrary number of following zeros, e.g.:

$$\text{lsbf}(6) = \mathcal{L}(0110^*) \text{ and } \text{lsbf}(0) = \mathcal{L}(0^*).$$

(a) Construct and compare the minimal DFAs representing all even natural numbers $n \in \mathbb{N}_0$ w.r.t. the unary encoding (i.e., $n \mapsto a^n$), the msbf encoding, and the lsbf encoding.

(b) Consider the following FA $\mathcal{A}$ over the alphabet $\{00, 01, 10, 11\}$:



W.r.t. the msbf encoding, we may interpret any word $w \in \{00, 01, 10, 11\}^*$ as a pair of natural numbers $(X(w), Y(w)) \in \mathbb{N}_0 \times \mathbb{N}_0$. *Example*: (Underlined letters correspond to $Y(w)$.)

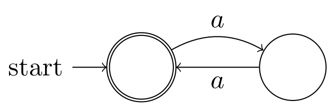
$$w = (00)^k001011 \to (0\underline{0})^k0\underline{0}1\underline{0}1\underline{1} \to (0^k011, 0^k001) \to (3, 1) = (X(w), Y(w))$$

- Find constants $a, b \in \mathbb{Z}$ such that $aX(w) + bY(w) = 0$ for all $w \in \mathcal{L}(\mathcal{A})$.

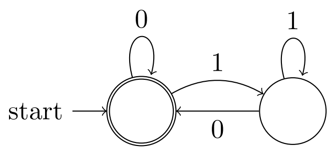
(c) Construct the minimal DFA representing the language $\{w \in \{0, 1\}^* \mid \mathrm{msbf}^{-1}(w)$ is divisible by $3\}$.

Solution:

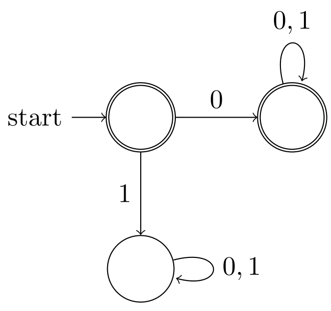
(a)
- Unary encoding:



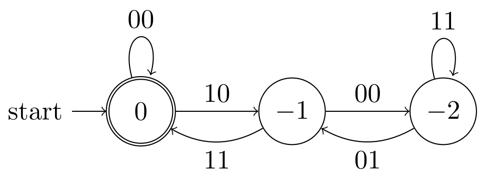
- MSBF encoding:



- LSBF encoding:



(b) We label the states as follows:



We show by induction on the length of a word $w \in \Sigma^*$ that for $\delta(0, w) = q$ we have $3Y(w) - X(w) = q$. Obviously, this will then imply that $3Y(w) - X(w) = 0$ for all $w \in \mathcal{L}(\mathcal{A})$.

Let l denote the length of a word w (w.r.t. the alphabet $\{00, 01, 10, 11\}$).

- $l = 0$:

  We have $w = \varepsilon$ and both $\delta(0, \varepsilon) = 0$ and $X(w) = 0 = Y(w)$.

- $l \to l + 1$:

  We may write w as uab with $a, b \in \{0, 1\}$. Note that we then have

  $$X(w) = 2X(u) + a \text{ and } Y(w) = 2Y(w) + b.$$

  Assume that $\delta(0, w) =: q$ is defined, otherwise there is nothing to show. Then also $\delta(0, u) =: q'$ is defined. By induction hypothesis we have $3Y(u) - X(u) = q'$.

  – Assume $q = 0 \wedge q' = 0$:

  Then $a = 0, b = 0$ has to hold, i.e., $X(w) = 2X(u)$ and $Y(w) = 2Y(u)$. Hence,

  $$3Y(w) - X(w) = 3 \cdot 2Y(u) - 2X(u) = 2(3Y(u) - X(u)) = 2 \cdot q' = 2 \cdot 0 = 0 = q.$$

  - Assume $q = 0 \wedge q' = -1$.

    Then $a = 1, b = 1$, and $X(w) = 2X(u) + 1$ and $Y(w) = 2Y(u) + 1$ subsequently follow, leading to:

    $$3Y(w) - X(w) = 3 \cdot (2Y(u) + 1) - (2X(u) + 1) = 2(3Y(u) - X(u)) + 2 = 2 \cdot q' + 2 = 2 \cdot (-1) + 2 = 0 = q.$$

  - Similarly, the remaining cases follow.

  One can even show that $\mathcal{A}$ accepts exactly those words w with $3Y(w) - X(w) = 0$. For this, let $w = uab$ be a word such that $\delta(0, u)$ is defined, but $\delta(0, w)$ is undefined, i.e., is the rejecting state.

  Assume $\delta(0, u) = 0$. Then either $ab = 01$ or $ab = 11$.

  - We only consider the case $ab = 01$, as the case $ab = 11$ is quite similar. Then by our preceding result we have $3Y(u) - X(u) = 0$, which leads to

    $$3Y(w) - X(w) = 3(2Y(u) + 1) - 2X(u) = 3.$$

    Assume we add a letter xy to w. In order to calculate $3Y(wxy) - X(wxy)$ we double the value $3Y(w) - X(w)$ and then add a number from $\{-1, 0, 2, 3\}$. Hence, $5 = 2(3Y(w) - X(w)) - 1 \leq 3Y(wxy) - X(wxy) \leq 2(3Y(w) - X(w)) + 3 = 9$, i.e., no matter how we extend w by some word $u \in \{00, 01, 10, 11\}^*$, we never will be able to obtain $3Y(wu) - X(wu)$.

  Similarly, one shows for the other cases of $\delta(0, u)$ that all words leading to the rejecting state cannot satisfy the linear equation $3y - x = 0$.

(c) The idea is that the state reached after reading the word u corresponds to the remainder of the number represented by u when dividing by 3.

We therefore take as states $\{0, 1, 2\}$ with 0 the initial state and define

$$\delta(q, a) = 2q + a \pmod 3.$$

This yields the automaton:

$$\begin{tikzcd}
\text{start} \arrow[r] & \boxed{0} \arrow[loop above, "0"] \arrow[r, bend left, "1"] & 1 \arrow[l, bend left, "1"] \arrow[r, bend left, "0"] & 2 \arrow[l, bend left, "0"] \arrow[loop above, "1"]
\end{tikzcd}$$

Obviously, this automaton has to be minimal as two different states encode two different remainder classes. Note that we also obtain this automaton from the one of (b) by forgetting (projecting) the y-component.

Exercise 2.3

In the lecture you have seen the definition of the Myhill-Nerode relation $\sim_L$ for a given language $L \subseteq \Sigma^*$:

For two words $x, y \in \Sigma^*$ we write $x \sim_L y$ iff $xv \in L \Leftrightarrow yv \in L$ holds for all $v \in \Sigma^*$.

(a) Determine the equivalence classes of $\sim_L$ w.r.t. the following languages over $\Sigma = \{a, b\}$:

- $L_1 := \mathcal{L}((ab + ba)^*)$,
- $L_2 := \mathcal{L}((aa)^*)$,
- $L_3 := \{w \in \{a, b\}^* \mid$ the number of occurrences of ab and ba in w is the same $\}$.
- $L_4 := \{a^n b^n c^n \mid n \geq 0\}$.

(b) In the definition of $\sim_L$ we compare two given words by appending all possible words. Instead of appending, we may also prepend. Consider therefore the binary relation $\sim^L$ on Σ^* defined by

For two words $x, y \in \Sigma^*$ we write $x \sim^L y$ iff $ux \in L \Leftrightarrow uy \in L$ holds for all $u \in \Sigma^*$.

- Determine the equivalence classes of $\sim^{L_1}$ and $\sim^{L_2}$.
- Show that L is regular iff $\sim^L$ has only finitely many equivalence classes.
- Is the number of equivalence classes of $\sim_L$ equal to the one of $\sim^L$?

(c) Finally, we may compare two words x, y also by appending and prepending arbitrary words, i.e., define the relation $\equiv_L$ as follows:

For two words $x, y \in \Sigma^*$ we write $x \equiv_L y$ iff $uxv \in L \Leftrightarrow uzv \in L$ holds for all $u, v \in \Sigma^*$.

For $x \in \Sigma^*$ let $[x]_L$ denote the equivalence class of x w.r.t. $\equiv_L$, i.e., $[x]_L = \{t \in \Sigma^* \mid x \equiv_L y\}$. We write $\Sigma^*/\equiv_L$ for the set of equivalence classes.

- How does $\equiv_L$ relate to $\sim_L$, resp. $\sim^L$?
- Determine the equivalence classes of $\equiv_{L_2}$.
- Show that the following multiplication on $\Sigma^*/\equiv_L$ is well-defined, associative and has $[\varepsilon]_L$ as its neutral element:

$$[w]_L \cdot [w']_L := [ww']_L.$$

Remark: $\Sigma^*/\equiv_L$ is called the syntactic monoid of L.

Solution:

(a) *Reminder*:

- For $L \subseteq \Sigma^*$, the relation $\sim_L$ on Σ^* was defined by

$$x \sim_L y \text{ iff } \forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L.$$

- Let $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ be a DFA. Set $P(q) = \{x \in \Sigma^* \mid \delta(q_I, x) = q\}$ for $q \in Q$.

  Obviously, for $x, y \in P(q)$ we have for any word $v \in \Sigma^*$ that

$$\delta(q_{\text{initial}}, xv) = \delta(\delta(q_{\text{initial}}, x), v) = \delta(q, v) = \delta(\delta(q_{\text{initial}}, y), v) = \delta(q_{\text{initial}}, yv).$$

  as the automaton is deterministic, i.e.,

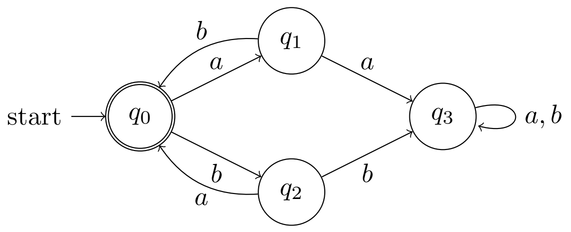
$$x, y \in P(q) \Rightarrow x \sim_L y.$$

  Hence, the partition $\{P(q) \mid q \in Q\}$ is a refinement of $\Sigma^*/\sim_L$. So, if $\mathcal{A}$ is also minimal, then we even have

$$x, y \in P(q) \Leftrightarrow x \sim_L y.$$

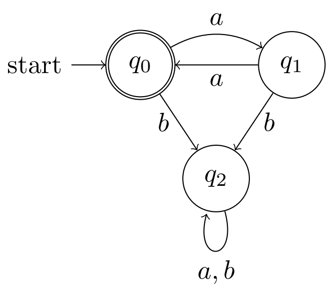
Hence, for calculating the equvialence classes of a regular language L it suffices to construct a minimal DFA for L and obtain from it the languages $P(q)$.

- L_1:



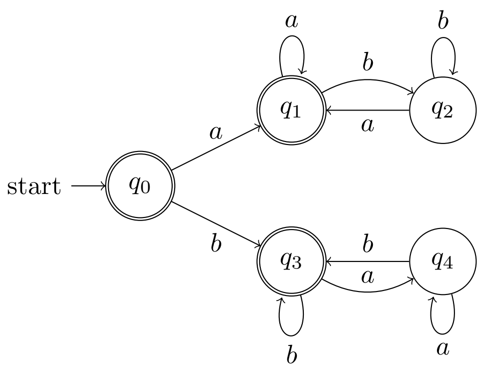
  $P(q_0) = \mathcal{L}((ab + ba)^*)$, $P(q_1) = \mathcal{L}((ab + ba)^* a)$, $P(q_2) = \mathcal{L}((ab + ba)^* b)$ and $P(q_3) = \mathcal{L}((ab + ba)^*(aa + bb))$.

- L_2:



  $P(q_0) = \mathcal{L}((aa)^*)$, $P(q_1) = \mathcal{L}((aa)^* a))$ and $P(q_2) = \mathcal{L}((aa)^*(b + ab))$.

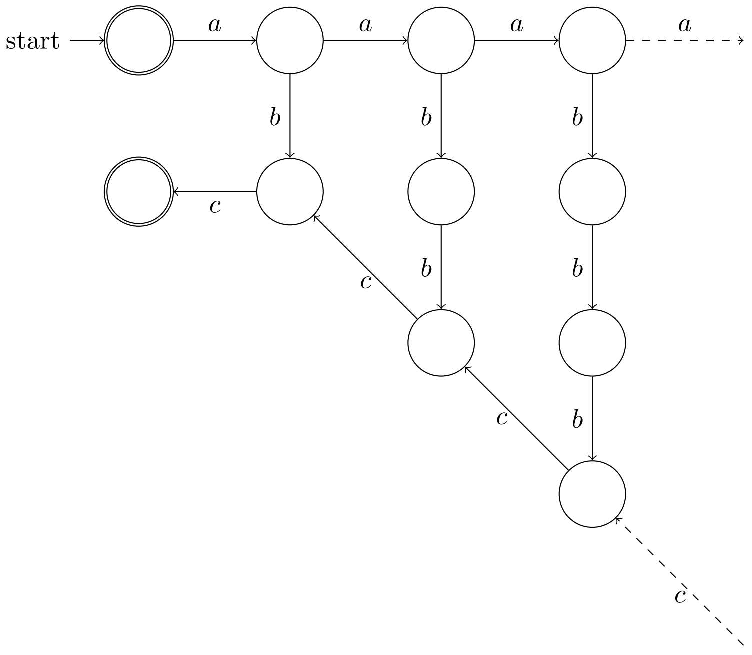
- L_3:



It's left to the reader to determine the equivalence classes from the DFA.

- L_4:

  Although L_4 is context-sensitive, but neither context-free nor regular, the same idea as for the preceding languages can be used, albeit one has to consider an automaton with a infinite number of states (the rejecting state isn't shown here):



  One easily checks that any two states are inequivalent. We obtain the equivalence classes:

  $\{a^k\}$ for $k \geq 0$, $\{a^k b^l\}$ for $k > l \geq 1$, $\{a^{k+l} b^{k+l} c^l \mid l \geq 0\}$ for $k \geq 1$, and the complement of the union of these.

(b) For $w \in \Sigma^*$ let w^R be its reverse, i.e., the word we obtain when reading w from right to left, e.g., $(abb)^R = bba$. For $L \subseteq \Sigma^*$ we then set $L^R := \{w^R \mid w \in L\}$.

It is well-known that L is regular if and only if L^R is regular (exercise!).

Consider the definition of $\sim^L$ now:

$$\begin{array}{lll} x \sim^L y & \text{iff} & \forall u \in \Sigma^* ux \in L \Leftrightarrow uy \in L \\ & \text{iff} & \forall u \in \Sigma^* (ux)^R \in L^R \Leftrightarrow (uy)^R \in L^R \\ & \text{iff} & \forall u \in \Sigma^* x^R u^R \in L^R \Leftrightarrow y^R u^R \in L^R \\ & \text{iff} & \forall v \in \Sigma^* x^R v \in L^R \Leftrightarrow y^R v \in L^R \\ & \text{iff} & \forall v \in \Sigma^* x^R v \in L^R \Leftrightarrow y^R v \in L^R \\ & \text{iff} & x^R \sim_{L^R} y^R. \end{array}$$

The crucial point here is that u^R is any possible word as we consider all $u \in \Sigma^*$ which allows us to replace u^R and u by v.

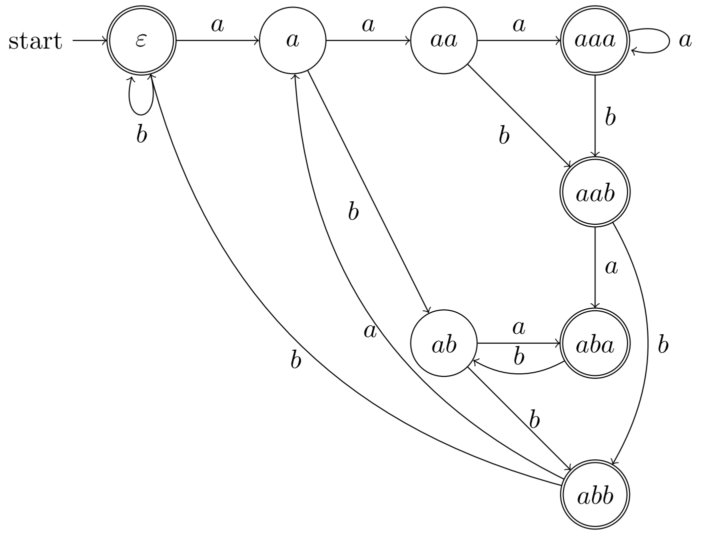
This means that we can obtain the equivalence classes of $\sim^L$ from those of $\sim_{L^R}$ by simply reversing the members of the equivalence classes, and we already know how to obtain the equivalence classes of $\sim_{L^R}$ (see (a)).

In both cases, L_1 and L_2, $L_i^R = L_i$, so we have already obtained the sought equivalence classes in (a).

It also immediately follows that $\sim^L$ is finite iff L is regular, as $|\sim^L| = |\sim_{L^R}|$ and we know that (i) L is regular iff L^R is regular, and (ii) L^R is regular iff $|\sim_{L^R}|$ is finite.

Finally, we show that in general $\sim_L$ and $\sim^L$ do not need to have the same number of classes: Let $L \subseteq \Sigma^*$ consist of those words whose third letter is an a. It is easy to see that a minimal DFA for L consists of $4+1$ states; four states are required to count how many letters have already been read, while the fifth state is the rejecting state.

On the other hand, L^R consists of all words whose third last letter is an a. A corresponding minimal DFA has to remember the last a seen within a window of three letters, leading to the equivalence classes $[\varepsilon]$, $[a]$, $[aa]$, $[aaa]$, $[aab]$, $[ab]$, $[aba]$, $[abb]$:



(c)
- Obviously, $x \equiv_L y$ implies both $x \sim_L y$ and $x \sim^L y$.

  But in general, $\equiv_L$ has more equivalence classes than $\sim_L$ or $\sim^L$. Consider for example L_1: We have both $\varepsilon \sim_{L_1} ab$ and $\varepsilon \sim^{L_1} ab$, but $\varepsilon \not\equiv_{L_1} ab$ as $a\varepsilon b \in L_1$, but $a(ab)b \notin L_1$. (See also exercise 3.2.)
- In the case of L_2 one easily checks $x \sim_{L_2} y$ iff $x \equiv_{L_2} y$
- It remains to show that the defined multiplication on $\Sigma^* / \equiv_L$ is independent of the representative taken from the equivalence class.

  Choose $x, y, x', y' \in \Sigma^*$ s.t. $x \equiv_L x'$ and $y \equiv_L y'$. We have to show that $xy \equiv_L x'y'$:
  - $x \equiv_L x'$ implies $xy \equiv_L x'y$. (Set $v = yv'$ in the definition with $v' \in \Sigma^*$).
  - $y \equiv_L y'$ implies $x'y \equiv_L x'y'$. (Set $u = u'x$ in the definition with $u' \in \Sigma^*$.)
  - As $\equiv_L$ is a equivalence relation, it is transitive, and we obtain $xy \equiv_L x'y'$.

Exercise 2.4

Let Σ be an alphabet. The set $\mathbb{S}_\Sigma$ of *star-free* expression over Σ is inductively defined:

$$\begin{array}{lcl} S^0 & := & \Sigma \cup \{\varepsilon, \emptyset\} \\ S^{k+1} & := & \{(\phi + \psi), (\phi \cdot \psi), \overline{\phi}, (\phi \cap \psi) \mid \phi, \psi \in S^k\} \cup S^k \\ \mathbb{S}_\Sigma & := & \bigcup_{k \in \mathbb{N}} S^k. \end{array}$$

The language $\mathcal{L}(\rho)$ represented by a star-free expression $\rho \in \mathbb{S}_\Sigma$ is defined as expected:

$$\mathcal{L}(\rho) := \begin{cases} \emptyset & \text{if } \rho = \emptyset & \mathcal{L}(\phi) \cup \mathcal{L}(\psi) & \text{if } \rho = (\phi + \psi) \\ \{\rho\} & \text{if } \rho \in \Sigma \cup \{\varepsilon\} & \mathcal{L}(\phi) \cdot \mathcal{L}(\psi) & \text{if } \rho = (\phi \cdot \psi) \\ \Sigma^* \setminus \mathcal{L}(\phi) & \text{if } \rho = \overline{\phi} & \mathcal{L}(\phi) \cap \mathcal{L}(\psi) & \text{if } \rho = (\phi \cap \psi). \end{cases}$$

To avoid parentheses, we assume that concatenation has the highest priority, followed by intersection, then addition. Some examples:

$$\mathcal{L}(\overline{\emptyset}) = \Sigma^* \setminus \mathcal{L}(\emptyset) = \Sigma^*, \quad \mathcal{L}(\overline{a} \cap (a + ab)) = \{ab\}.$$

(a) Show that for every star-free expression $\phi \in \mathbb{S}_\Sigma$ it holds that $\mathcal{L}(\phi)$ is a regular languages over Σ.

(b) Give a star-free expression for the regular language $\mathcal{L}((ab)^*)$ for $\Sigma = \{a, b\}$.

Remark: There is no star-free expression for the language $\mathcal{L}((aa)^*)$!

Solution:

(a) We claim that

$$\mathcal{L}(a\overline{\emptyset}b \cap \overline{\overline{\emptyset}(aa+bb)\overline{\emptyset}}) \overset{!}{=} \mathcal{L}((ab)(ab)^*).$$

Note that $\mathcal{L}(\overline{\emptyset}) = \Sigma^* = \mathcal{L}((a+b)^*)$. So we may write:

$$\mathcal{L}(a\overline{\emptyset}b \cap \overline{\overline{\emptyset}aa\overline{\emptyset} + \overline{\emptyset}bb\overline{\emptyset}}) = \mathcal{L}(a(a+b)^*b) \setminus \mathcal{L}((a+b)^*(aa+bb)(a+b)^*).$$

- This follows from a straightforward induction on the term structure using that regular languages are closed under union, intersection and complement.
- $\mathcal{L}(a(a+b)^*b) \setminus \mathcal{L}((a+b)^*(aa+bb)(a+b)^*) \supseteq \mathcal{L}((ab)(ab)^*)$:

  Consider $w = (ab)^k$ for some $k \geq 1$. Clearly, $w \in \mathcal{L}(a(a+b)^*b)$. Further, neither aa nor bb appear as subwords in w, so $w \notin \mathcal{L}(\Sigma^*(aa+bb)\Sigma^*)$.
- $\mathcal{L}(a(a+b)^*b) \setminus \mathcal{L}((a+b)^*(aa+bb)(a+b)^*) \subseteq \mathcal{L}((ab)(ab)^*)$:

  Choose some $w \in \mathcal{L}(a(a+b)^*b) \setminus \mathcal{L}((a+b)^*(aa+bb)(a+b)^*)$. (Note that this language is not empty as it contains the word ab.) Obviously, $w \neq \varepsilon$.

  One now shows by induction on the length of w, that the letters a and b have to alternate in w and that w has to be of even length (as it starts with an a, but ends with a b). So, $w \in \mathcal{L}((ab)(ab)^*)$.

It follows that

$$\mathcal{L}((ab)^*) = \mathcal{L}(\varepsilon + a\overline{\emptyset}b \cap \overline{\overline{\emptyset}(aa+bb)\overline{\emptyset}}).$$