Prof. Dr. Javier Esparza Garching b. München, den 10.08.17

Klausur

Einführung in die theoretische Informatik

Sommer-Semester 2017

Beachten Sie:

- Soweit nicht anders angegeben, ist stets eine Begründung bzw. der Rechenweg anzugeben!
- Die Klausur besteht aus 8 Aufgaben auf 6 Seiten. Insgesamt können Sie 40 Punkte erreichen.
- $|w|_{\mathsf{a}}$ für $w \in \Sigma^*$ und $\mathsf{a} \in \Sigma$ bezeichnet die Anzahl der as in w, z.B. $|\mathsf{aabab}|_{\mathsf{a}} = 3$.
- Nichtbeachtung der aufgabenspezifischen Hinweise kann zu Punktabzug führen.

Viel Erfolg !

Aufgabe 1 (Quiz zu Regulären und Kontextfreien Sprachen) (5 Punkte)

Begründen Sie, ob die folgenden Aussagen jeweils korrekt oder inkorrekt sind. Falls die Aussage korrekt ist, beziehen Sie sich dazu auf die entsprechenden Ergebnisse aus der Vorlesung. Falls die Aussage inkorrekt ist, geben Sie bitte ein passendes Gegenbeispiel an.

(a) Seien A und B nicht-reguläre Sprachen. Dann ist $\mathsf{A} \cup \mathsf{B}$ nicht regulär.

Lösungsskizze: Nicht korrekt. Setze $\mathsf{A} = \{\mathsf{a}^n\mathsf{b}^n \mid n \geq 0\}$ und $\mathsf{B} = \overline{\mathsf{A}}$. A ist nach Vorlesung nicht regulär und, da reguläre Sprachen unter Komplement abgeschlossen sind, auch nicht B. Es somit gilt $\mathsf{A} \cup \mathsf{B} = \mathsf{A} \cup \overline{\mathsf{A}} = \Sigma^*$. Damit ist die Vereinigung aber regulär.

(b) Seien A eine reguläre, nicht-leere und B eine nicht-reguläre Sprache. Dann ist AB nicht regulär.

Lösungsskizze: Nicht korrekt. Setze $\mathsf{A} = \Sigma^*$ und $\mathsf{B} = \{\mathsf{a}^n\mathsf{b}^n \mid n \geq 0\}$. Dann ist $\mathsf{AB} = \Sigma^*$ und damit regulär, obwohl A regulär und B nicht regulär ist.

(c) Sei $\mathsf{A} \subseteq \Sigma^*$ eine reguläre Sprache über dem Alphabet $\Sigma = \{\mathsf{a}, \mathsf{b}\}$. Dann gibt es mindestens eine Myhill-Nerode-Äquivalenzklasse K bezüglich $\equiv_{\mathsf{A}}$ mit $|K| = \infty$.

Lösungsskizze: Korrekt. Da A regulär ist, gibt es nur endlich viele Äquivalenzklasse, die Σ^* partionieren. Da aber Σ^* unendlich groß ist, muss es eine Klasse K mit $|K| = \infty$ geben.

(d) Es gibt keinen minimalen DFA $\mathsf{D} = (\mathsf{Q}, \Sigma, \delta, \mathsf{q}_0, \mathsf{F})$ mit $|\mathsf{Q}| = 2$ und $|\mathsf{F}| = 0$.

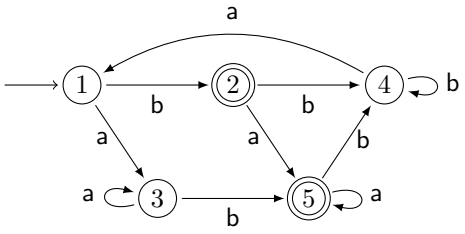
Lösungsskizze: Korrekt. Für jeden DFA D mit $|\mathsf{F}| = 0$ gilt: $\mathsf{L}(\mathsf{D}) = \emptyset$. Der minimiale DFA für $\emptyset$ hat aber nur genau einen Zustand. Somit ist kein DFA mit diesen Eigenschaften minimal.

(e) Für jeden PDA P gibt es einen PDA P' mit $\mathsf{L}_\varepsilon(\mathsf{P}) \setminus \{\varepsilon\} = \mathsf{L}_\varepsilon(\mathsf{P}')$, der keine ε-Transitionen besitzt.

Lösungsskizze: Korrekt. Nach Vorlesung gibt es zu jedem PDA P eine Grammatik G, die die gleiche Sprache beschreibt. Man kann G unter Erhaltung der Sprachgleichheit bis auf das leere Wort ε in G' in Greibach-Normalform transformieren. Dazu gibt es nach Vorlesung einen PDA P', der die gleiche Sprache akzeptiert und keine ε-Transitionen enthält. *Zusammengefasst:* $\mathsf{P} \to \mathsf{G} \to \mathsf{G}'$ in Greibach Normalform $\to$ P' ohne ε-Transitionen

Aufgabe 2 (Minimierung) (7 Punkte)

Betrachten Sie den folgenden DFA D über dem Alphabet $\Sigma = \{\mathsf{a}, \mathsf{b}\}$:



(a) Minimieren Sie den DFA D. Geben Sie außerdem die Tabelle der kürzesten unterscheidenden Wörter an, die Sie zur Minimierung des Automaten verwenden.

(b) Bestimmen Sie die Anzahl der Myhill-Nerode-Äquivalenzklassen der Sprache $\mathsf{L}(\mathsf{D})$.

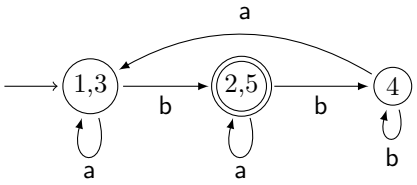
(c) Geben Sie zu jeder Klasse einen regulären Ausdruck an, der alle Wörter der Klasse beschreibt. Eine Begründung hierfür ist nicht notwendig.

Lösungsskizze:

(a) Die Tabelle zur Minimierung mit den kürzesten unterscheidenden Wörtern:

	1	2	3	4	5
1	–	–	–	–	–
2	ε	–	–	–	–
3	$\equiv$	ε	–	–	–
4	b	ε	b	–	–
5	ε	$\equiv$	ε	ε	–

Der Minimalautomat:



(b) Anzahl der Myhill-Nerode-Klassen: 3

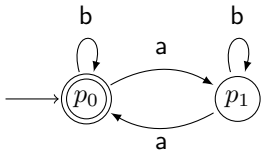
(c) Zuordnung von Nummer zu Äquivalenzklasse und regulärem Ausdruck

Nummer	Äquivalenzklasse	regulärer Ausdruck
1,3	$[\varepsilon]$	$(\mathsf{a}^*\mathsf{ba}^*\mathsf{bb}^*\mathsf{a})^*\mathsf{a}^*$
2,5	$[\mathsf{b}]$	$(\mathsf{a}^*\mathsf{ba}^*\mathsf{bb}^*\mathsf{a})^*\mathsf{a}^*\mathsf{ba}^*$
4	$[\mathsf{bb}]$	$(\mathsf{a}^*\mathsf{ba}^*\mathsf{bb}^*\mathsf{a})^*\mathsf{a}^*\mathsf{ba}^*\mathsf{bb}^*$

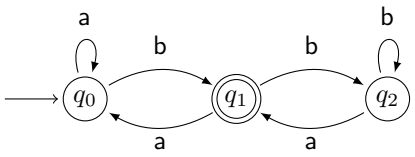
Aufgabe 3 (Sprachdifferenz) (3 Punkte)

Konstruieren Sie unter Verwendung der Produktkonstruktion einen DFA D_3 mit $\mathsf{L}(\mathsf{D}_3) = \mathsf{L}(\mathsf{D}_1) \setminus \mathsf{L}(\mathsf{D}_2)$.

D_1:



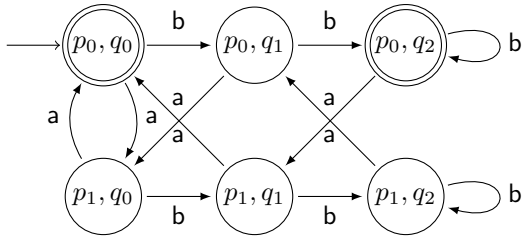
D_2:



Hinweis: Beschriften Sie die Zustände mit Paaren von Zuständen von D_1 und D_2.

Lösungsskizze:

D_3:



Aufgabe 4 (Reguläre Sprachen) (5 Punkte)

Sei Σ ein Alphabet und RE die Menge aller regulären Ausdrücke über Σ.

(a) Geben Sie eine rekursive Prozedur $f\colon \mathsf{RE} \to \mathbb{B}$ an, so dass $f(r) \Leftrightarrow \varepsilon \in \mathsf{L}(r)$.

(b) Sei $\mathsf{L} \subseteq \Sigma^*$ eine Sprache über dem Alphabet Σ. Für jedes $x \in \Sigma$ definieren wir $\mathsf{L}_x \subseteq \Sigma^*$ als:

$$\mathsf{L}_x := \{w \in \Sigma^* \mid wx \in \mathsf{L}\}$$

Beispiel: Sei $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ und $\mathsf{L} = \{\mathsf{acb}, \mathsf{bbc}, \mathsf{b}, \mathsf{ba}\}$. Dann ist $\mathsf{L}_\mathsf{b} = \{\varepsilon, \mathsf{ac}\}$.

Geben Sie eine rekursive Prozedur $g\colon \Sigma \times \mathsf{RE} \to \mathsf{RE}$ an, so dass $\mathsf{L}(g(x, r)) = \mathsf{L}(r)_x$ für jeden regulären Ausdruck r und jedes $x \in \Sigma$.

Hinweis: Für die Definition von g dürfen Sie f verwenden.

Lösungsskizze:

$$f(\emptyset) = \mathsf{false}$$
$$f(\varepsilon) = \mathsf{true}$$
$$f(a) = \mathsf{false}$$
$$f(\alpha|\beta) = f(\alpha) \vee f(\beta)$$
$$f(\alpha\beta) = f(\alpha) \wedge f(\beta)$$
$$f(\alpha^*) = \mathsf{true}$$

$$g(x, \emptyset) = \emptyset$$
$$g(x, \varepsilon) = \emptyset$$
$$g(x, a) = \begin{cases} \varepsilon & \text{falls } x = a \\ \emptyset & \text{sonst} \end{cases}$$
$$g(x, \alpha|\beta) = g(x, \alpha)|g(x, \beta)$$
$$g(x, \alpha\beta) = \begin{cases} g(x, \alpha)|\alpha g(x, \beta) & \text{falls } f(\beta) \\ \alpha g(x, \beta) & \text{sonst} \end{cases}$$
$$g(x, \alpha^*) = \alpha^* g(x, \alpha)$$

Aufgabe 5 (Kontextfreie Sprachen) (5 Punkte)

Betrachten Sie die Sprache $\mathsf{L} = \{\mathsf{a}^i\mathsf{b}^j\mathsf{c}^k \mid i = j + 2k \wedge i, j, k \geq 0\}$ über dem Alphabet $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$.

(a) Konstruieren Sie einen DPDA P mit $\mathsf{L} = \mathsf{L}_F(\mathsf{P})$, d.h. P akzeptiert L mit Endzuständen.

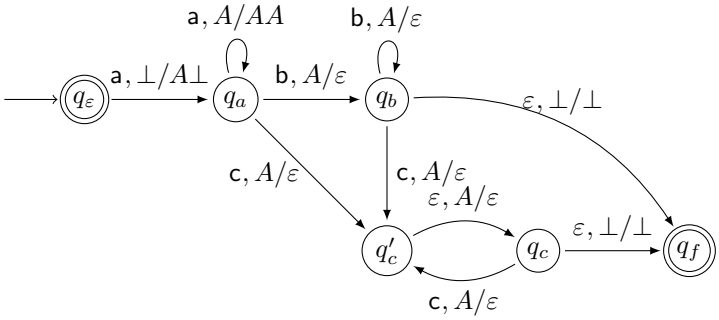
(b) Konstruieren Sie eine CFG G mit $\mathsf{L} = \mathsf{L}(\mathsf{G})$.

Hinweise:

- Achten Sie darauf, dass Ihr PDA P deterministisch ist und geben Sie δ des DPDAs graphisch an.
- Verwenden Sie für den DPDA P maximal 2 Stacksymbole und 8 Zustände. Beachten Sie, dass unsere Lösung weniger Zustände verwendet.
- Verwenden Sie für G maximal 3 Variablen und maximal 5 Produktionen. Beachten Sie, dass unsere Lösung weniger Variablen und Produktionen verwendet.
- *Achtung:* Stellen Sie sicher, dass Ihr DPDA bzw. Ihre Grammatik folgende Wörter erkennt $\varepsilon, \mathsf{ab}, \mathsf{aac}, \mathsf{aaabc}$ und folgende Wörter nicht erkennt $\mathsf{a}, \mathsf{abc}, \mathsf{caa}$. Sonst wird die Aufgabe mit 0 Punkten bewertet.

Lösungsskizze:

DPDA $\mathsf{P} = (\{q_\varepsilon, q_a, q_b, q_c, q_c', q_f\}, \{\bot, A\}, \delta, \bot, \{q_\varepsilon, q_f\})$ mit δ:



CFG $\mathsf{G} = (\{\mathsf{S}, \mathsf{S}'\}, \Sigma, P, \mathsf{S})$ mit P:

$$\mathsf{S} \to \mathsf{aaSc} \mid \mathsf{S}' \qquad \mathsf{S}' \to \mathsf{aS}'\mathsf{b} \mid \varepsilon$$

Aufgabe 6 (Pumping-Lemma für Kontextfreie Sprachen) (5 Punkte)

Zeigen Sie mithilfe des Pumping-Lemmas für kontextfreie Sprachen, dass $\mathsf{L} = \{\mathsf{a}^i\mathsf{b}^j\mathsf{c}^k \mid i = j \cdot k \wedge i, j, k \geq 0\}$ über dem Alphabet $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ nicht kontextfrei ist.

Lösungsskizze: Wir nehmen an, dass L kontextfrei ist und führen diese Annahme zum Widerspruch.

Sei $n \geq 1$ eine Pumping-Lemma-Zahl für die kontextfreie Sprache L. Wir wählen $z = \mathsf{a}^{n^2}\mathsf{b}^n\mathsf{c}^n$. Somit gilt $z \in \mathsf{L}$ und $|z| \geq n$. Gemäß Pumping-Lemma gibt es dann eine Zerlegung $z = uvwxy$ mit Wörtern $u, v, w, x, y \in \Sigma^*$ und den folgenden Eigenschaften:

$$(1)\ vx \neq \varepsilon \qquad (2)\ |vwx| \leq n \qquad (3)\ \forall i \in \mathbb{N}_0.\ uv^iwx^iy \in \mathsf{L}$$

Wir unterscheiden die folgenden zwei Fälle:

- $|vx|_\mathsf{a} > 0$:
  Wegen (2) gilt $|vx|_\mathsf{c} = 0$. Sei $i = |vx|_\mathsf{a}$ und sei $j = |vx|_\mathsf{b}$. Aus (3) folgt $uwy \in \mathsf{L}$ und somit muss es auch ein Lösung für $n^2 - i = (n - j) \cdot n$ geben. Nach Umstellung erhalten wir die Gleichung $i = j \cdot n$, für die es keine Lösung unter der Randbedingung $1 \leq i + j \leq n$ (1,2) gibt. Wiederspruch.
- $|vx|_\mathsf{a} = 0$:
  Sei $i = |vx|_\mathsf{b}$ und sei $j = |vx|_\mathsf{c}$. Aus (3) folgt $uv^2wx^2y \in \mathsf{L}$ und somit muss es auch ein Lösung für $n^2 = (n + i) \cdot (n + j)$ geben. Nach Umstellung erhalten wir die Gleichung $0 = i \cdot n + j \cdot n + i \cdot j$, für die es keine Lösung unter der Randbedingung $1 \leq i + j \leq n$ (1,2) gibt. Wiederspruch.

Da jeder Fall zu einem Widerspruch führt und die obigen Fälle alle möglichen Zerlegungen abdecken, kann die ursprüngliche Annahme nicht gelten. Also ist L nicht kontextfrei.

Aufgabe 7 (Quiz zu Entscheidbarkeit) (5 Punkte)

Begründen Sie, ob die folgenden Aussagen jeweils korrekt oder inkorrekt sind. Falls die Aussage korrekt ist, beziehen Sie sich dazu auf die entsprechenden Ergebnisse aus der Vorlesung. Falls die Aussage inkorrekt ist, geben Sie bitte ein passendes Gegenbeispiel an.

(a) Für jede entscheidbare Menge $\mathsf{A} \subseteq \{0,1\}^*$ gibt es eine TM, mit mehr als 314 Zuständen, die χ_{A} berechnet.

Lösungsskizze: Korrekt. Füge nicht erreichbare Zustände hinzu bis die Turing-Maschine mehr als 314 Zustände hat.

(b) Seien A und B Sprachen. Ist $\mathsf{A} \leq \mathsf{B}$ und A entscheidbar, dann ist auch B entscheidbar.

Lösungsskizze: Inkorrekt. Sei $\Sigma = \{0,1\}$, $\mathsf{A} = \{1\}$ und $\mathsf{B} = \mathsf{H}_0$. Wir verwenden die Funktion, die 1 auf ein Element von H_0 abbildet und alle anderen auf ein Element von $\overline{\mathsf{H}_0}$, als Reduktion von A auf H_0. Diese Funktion ist eindeutig berechenbar, A ist entscheidbar, H_0 aber nicht.

(c) Rekursiv aufzählbare Sprachen sind unter Komplement abgeschlossen.

Lösungsskizze: Inkorrekt. Eine Sprache ist rekursiv aufzählbar genau dann, wenn sie semi-entscheidbar ist. Das Halteproblem auf leerem Band H_0 ist semi-entscheidbar, $\overline{\mathsf{H}_0}$ ist aber nicht semi-entscheidbar.

(d) Sei $(\mathsf{L}_i)_{i\in\mathbb{N}}$ eine Familie entscheidbarer Sprachen, d.h. jedes L_i ist entscheidbar. Dann ist auch $\bigcup_{i\in\mathbb{N}} \mathsf{L}_i$ entscheidbar.

Lösungsskizze: Inkorrekt. $\mathsf{H}_0 \subseteq \{0,1\}^*$ und damit abzählbar. Daher gibt es eine surjektive Funktion $f\colon \mathbb{N} \to \mathsf{H}_0$. Wir setzen $\mathsf{L}_i \coloneqq \{f(i)\}$. Da endliche Sprachen entscheidbar sind, ist L_i für alle $i \in \mathbb{N}$ entscheidbar. da endliche Mengen regulär sind und somit entscheidbar. Es gilt $\bigcup_{i\in\mathbb{N}} \mathsf{L}_i = \mathsf{H}_0$ und damit nicht entscheidbar.

(e) Es gibt keine berechenbare Funktion $f\colon \mathsf{CFG} \times \mathsf{CFG} \to \mathbb{N}_0$, die im Fall $\mathsf{L}(\mathsf{G}_1) \not\subseteq \mathsf{L}(\mathsf{G}_2)$ die Länge eines kürzesten unterscheidenden Wortes $w \in \mathsf{L}(\mathsf{G}_1) \setminus \mathsf{L}(\mathsf{G}_2)$ zurückgibt: $f(\mathsf{G}_1, \mathsf{G}_2) = |w|$. Falls $\mathsf{L}(\mathsf{G}_1) \subseteq \mathsf{L}(\mathsf{G}_2)$ gilt, kann f auf einen beliebigen Wert abbilden.

Lösungsskizze: Korrekt. Sollte es eine solche Funktion geben, wäre $\mathsf{L}(\mathsf{G}_1) \subseteq \mathsf{L}(\mathsf{G}_2)$ mit folgendem Algorithmus entscheidbar: Berechne $c = f(\mathsf{G}_1, \mathsf{G}_2)$ und enumeriere alle Wörter w der Länge c und teste für jedes $w \in \mathsf{L}(\mathsf{G}_1)$ und $w \notin \mathsf{L}(\mathsf{G}_2)$. Falls es so ein w gibt, terminiere und gebe "Nein" aus. Falls es kein solches w gibt, dann gebe "Ja".

Aufgabe 8 (Reduktionen) (5 Punkte)

Mit DOUBLE-SAT bezeichnen wir im Folgenden das Problem, gegeben eine aussagenlogische Formel zu entscheiden, ob sie mindestens *zwei* erfüllbare Belegungen hat.

Formal: Sei $\mathcal{F}$ die Menge aller aussagenlogischen Formeln über einer unendlichen Menge $\mathcal{V}$ von Variablen. Sei $\sigma\colon \mathcal{V} \to \{0,1\}$ eine Belegung. Wie üblich wird σ konsistent mit der Semantik der Aussagenlogik auf $\mathcal{F} \to \{0,1\}$ erweitert. Dann können wir die zwei Mengen SAT und DOUBLE-SAT definieren:

$$\begin{aligned}\mathsf{SAT} &\coloneqq \{F \in \mathcal{F} \mid \exists \sigma\colon \mathcal{V} \to \{0,1\}.\ \sigma(F) = 1\}\\ \mathsf{DOUBLE\text{-}SAT} &\coloneqq \{F \in \mathcal{F} \mid \exists \sigma, \sigma'\colon \mathcal{V} \to \{0,1\}.\ \sigma(F) = 1 = \sigma'(F)\\ &\qquad \wedge \sigma(x) \neq \sigma'(x) \text{ für eine Variable } x, \text{ die in } F \text{ vorkommt.}\}\end{aligned}$$

Zeigen Sie die beiden folgenden Aussagen, indem Sie passende Reduktionen angeben und deren Korrektheit begründen. Geben Sie auch an, welche Reduktion zeigt, dass DOUBLE-SAT in NP liegt, und welche zeigt, dass DOUBLE-SAT NP-schwer (NP-hart) ist. Begründen Sie Ihre Antwort.

(a) DOUBLE-SAT $\leq_p$ SAT

Lösungsskizze:

Reduktion: $f(F) = F \wedge F[x \mapsto x' \mid x \in \mathcal{V}(F)] \wedge (\bigvee_{x \in \mathcal{V}(F)} x \oplus x')$, d.h. wir ersetzen jedes $x \in \mathcal{V}(F)$ mit einer neuen Variable x', die nicht in F vorkommt.

Korrektheit:

(1) $F \in$ DOUBLE-SAT. Dann existieren σ, σ' mit $\sigma(F) = 1 = \sigma''(F)$ und $\sigma(x) = 1 - \sigma'(x)$ für mindestens eine Variable x in F. Wir konstruieren dann ein $\sigma''(x) = \sigma(x)$ und $\sigma''(x') = \sigma'(x)$, d.h. σ'' verwendet die Belegung σ für die "alten" Variablen x und σ' für die kopierten Variablen x'. Somit gilt $\sigma''(f(F)) = 1$ und deshalb $f(F) \in$ SAT.

(2) $f(F) \in$ SAT, dann können wir aus der Belegung σ zwei verschiedene Belegungen σ', σ'' analog zu dem vorherigem Prinzip konstruiert werden. Somit $F \in$ DOUBLE-SAT.

Poly. Berechenbarkeit: F muss einmal kopiert werden und alle Variablen angepasst werden. Weiterhin muss die Disjunktion der Länge $\mathcal{O}(|\mathcal{V}(F)|)$ angehängt werden. Beides ist in polynomieller Laufzeit von einer DTM ausführbar.

(a) zeigt DOUBLE-SAT $\in$ NP, da SAT in NP liegt und wir mithilfe der Reduktion von (a) und der NTM für SAT eine NTM konstruieren können, die in polynomieller Zeit DOUBLE-SAT entscheidet.

(b) SAT $\leq_p$ DOUBLE-SAT

Lösungsskizze:

Reduktion: $f(F) = F \wedge (x \vee \neg x)$, wobei $x \in \mathcal{V}$ eine Variable ist, die nicht in F vorkommt.

Korrektheit:

(1) Sei $F \in$ SAT. Dann existiert ein σ mit $\sigma(F) = 1$. Wir setzen dann $\sigma'(x) = 0 = 1 - \sigma''(x)$ und $\sigma(y) = \sigma'(y) = \sigma''(y)$ für alle Variablen y aus F. Somit $\sigma'(f(F)) = 1 = \sigma''(f(F))$ und $f(F) \in$ DOUBLE-SAT.

(2) Sei $F \notin$ SAT. Dann existiert auch keine erfüllende Belegung für $f(F)$ und somit $f(F) \notin$ DOUBLE-SAT.

Poly. Berechenbarkeit: F muss einmal gelesen werden um ein *frisches* x zu finden und $\wedge(x \vee \neg x)$ muss an F angehängt werden. Beides ist in polynomieller Laufzeit von einer DTM ausführbar.

(b) zeigt DOUBLE-SAT ist NP-schwer, da laut Vorlesung SAT NP-schwer ist und somit DOUBLE-SATmindestens auch NP-schwer ist.