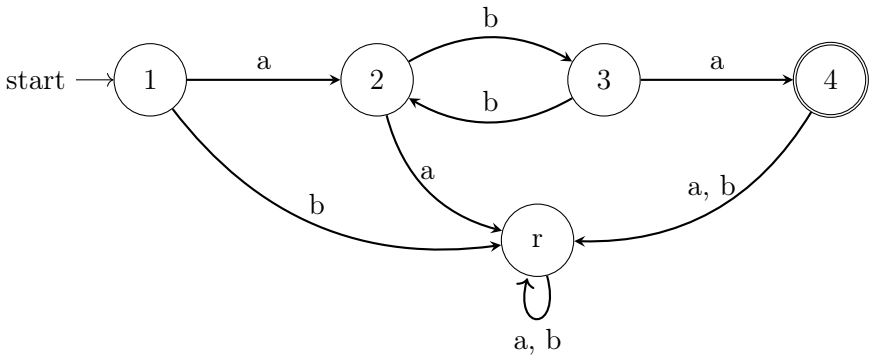
1) Der DEA für die Sprache $L = \{ab^n a | n \text{ ungerade}\}$ ist:



Die entsprechende TM dafür ist über die Übergangsfunktion definiert:

δ	$\vdash$	a	b	$\sqcup$
1	(1, $\vdash$, R)	(2, a, R)	-	-
2	-	-	(3, b, R)	-
3	-	(4, a, R)	(2, b, R)	-
4	-	-	-	(t, $\sqcup$, R)
r	-			

with $- := (r, *, R)$ r rejecting, t accepting.

2) *Lösung.*

```
while x3 ≠ 0 do
  x3 := x3 − 1
end;

while x1 ≠ 0 do
  x3 := x3 + 1;
  x1 := x1 − 1
end;

while x2 ≠ 0 do
  x3 := x3 + 1;
  x2 := x2 − 1
end
```

Python Script:

```
# modified decrease operation that doesn't return a negative number
def dec(x1):
```

```
    y = x1 - 1
    return 0 if y < 0 else y

def f(x1,x2,x3):
    while (x3 != 0):
        x3 = dec(x3)
    while(x1 != 0):
        x3 = x3 + 1
        x1 = dec(x1)
    while(x2 != 0):
        x3 = x3 + 1
        x2 = x2 - 1
    return (x1,x2,x3)
```

□

3) *Lösung.* Um zu beweisen, dass $L \leq_T M$ gilt muss eine Reduktionsfunktion $R : \Sigma^* \to \Sigma^*$ angegeben werden, die folgende Eigenschaften erfüllt:
   - R ist berechenbar (wird meist nur informell bewiesen, indem skizziert wird, wie ein möglicher Algorithmus für R aussieht)
   - R bildet jedes Wort in L auf ein Wort in M ab
   - R bildet jedes Wort nicht in L auf ein Wort nicht in M ab

   Nun zu den Konsequenzen einer Reduktion. Falls $L \leq_T M$ gilt, folgt insb.:
   - Falls M entscheidbar ist, so ist auch L entscheidbar
   - Falls M rekursiv aufzählbar ist, so ist auch L rekursiv aufzählbar

   Andersherum gilt auch:
   - Falls L unentscheidbar ist, so ist auch M unentscheidbar
   - Falls L nicht rekursiv aufzählbar ist, so ist auch M nicht rekursiv aufzählbar

   Als Eselsbrücke kann man $L \leq_T M$ lesen als "L ist leichter (oder genauso schwer) wie M". Wichtig ist insb. dass man, wenn man die Unentscheidbarkeit eines Problems M zeigen will, man ein bekanntes schweres Problem (z.B. das Halteproblem) *auf* M reduzieren muss, und nicht etwa M auf das Halteproblem (ein häufiger Fehler!).

   Nun zu der eigentlichen Aufgabe:

   a) Eine offensichtliche Möglichkeit ist $R(w) = 0w$. Dies ist offensichtlich berechenbar und offensichtlich hat auch $0w$ Länge 6 gdw. w Länge 5 hat.

      Da die Sprachen beide entscheidbar sind, können wir auch die gesamte "Arbeit" in der Reduktionsfunktion selbst machen und einfach eine Fallunterscheidung machen:

$$R'(w) = \begin{cases} u_1 & \text{falls } |w| = 5 \\ u_2 & \text{sonst} \end{cases}$$

wobei u_1 ein beliebiges Wort mit Länge 6 ist und u_2 ein beliebiges Wort mit Länge $\neq 6$, z.B. $u_1 = 000000$ und $u_2 = 0$.

b) Hier muss unsere Reduktionsfunktion für *alle* Eingaben ein Wort zurückgeben, das nicht in A liegt. Sei also u ein beliebiges Wort in $\{0,1\}^* \setminus \{A\}$. Dann ist die konstante Funktion $R(w) = u$ eine gültige Reduktionsfunktion.

   Dies geht natürlich nur, falls $A \neq \{0,1\}^*$ ist. Falls $A = \{0,1\}^*$ ist, gilt $\emptyset \leq_T A$ nicht.

c) Wir beschreiben im folgenden informell, wie unsere Reduktionsfunktion R vorgeht. Zunächst prüfen wir, ob unsere Eingabe w überhaupt ein gültiges Registermaschinenprogramm ist. Falls nicht, ist offensichtlich $w \notin A$ und wir geben einfach direkt wieder w zurück (da dann auch $w \notin B$ ist).

   Ansonsten prüfen wir, welche Register überhaupt in dem Programm vorkommen. Für jedes vorkommende Register i fügen wir dann vorne an das Programm eine Zuweisung der Form $R_i := 0$ hinzu.

   Dann ist klar, dass das ursprüngliche Programm terminiert wenn alle Register 0 sind gdw. unser modifiziertes Programm unabhängig vom initialen Registerinhalt terminiert.

   $B \leq_T A$ gilt übrigens nicht: das Problem, zu entscheiden, ob ein gegebenes Programm für alle Eingaben terminiert ist strikt schwieriger als zu entscheiden, ob es für eine bestimmte Eingabe terminiert. □