Algorithmisch unlösbare Probleme, Turing Maschinen, Registermaschinen, Komplexitätstheorie

Importancy	
Complete	☐
Notes	☐
Readings	☐

Entscheidbarkeit & Unentscheidbarkeit

> 💡 Ein Problem, welches nicht algorithmisch lösbar ist nennt man **unentscheidbar** andernfalls heißt das Problem **entscheidbar.**

Das **Halteproblem** beschreibt ein Programm, welches auf der Eingabe hält, heißt wenn das Programm nicht printen sollte, aber nur durch eine Bedingung und solange diese nicht Eintrifft wird nicht geprinted.

Postsches Korrespondenzproblem: Gegeben zwei gleich lange Listen von (nicht-leeren) Wörtern $w_1, w_2, \ldots, w_n$ und $x_1, x_2, \ldots, x_n$. Es werden die Indizes $i_1, i_2, \ldots, i_m$ gesucht, sodass $w_{i_1} w_{i_2} \ldots w_{i_m} = x_{i_1} x_{i_2} \ldots x_{i_m}$

Praktisches Beispiel:

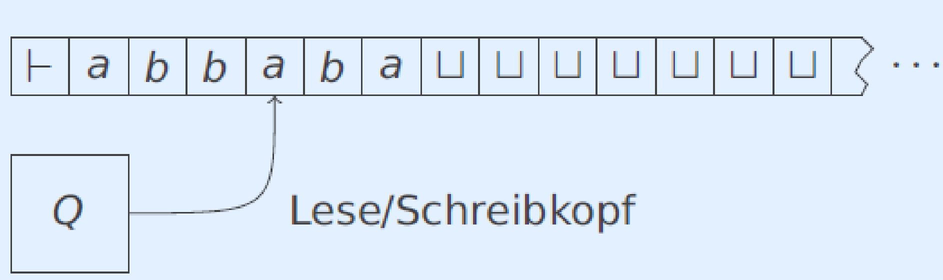
1) a) Wir nehmen an, dass ein $m > 0$ und Indizes existieren, sodass $x_{i_1} x_{i_2} \ldots x_{i_m} = y_{i_1} y_{i_2} \ldots y_{i_m}$. Wir unterscheiden folgende Fälle:
   i. Sei $i_1 = 1$, dann gilt für alle $m > 0$, dass $|x_{i_1} x_{i_2} \ldots x_{i_m}| \neq |y_{i_1} y_{i_2} \ldots y_{i_m}|$, d.h $x \neq y$.
   ii. Sei $i_1 = 2$, dann gilt $101 x_{i_2} \ldots x_{i_m} \neq 100 y_{i_2} \ldots y_{i_m}$.
   iii. Sei $i_1 = 3$, dann gilt $11000 x_{i_2} \ldots x_{i_m} \neq 101 y_{i_2} \ldots y_{i_m}$.
   Unabhängig von der Wahl für i_1 stimmen die Zeichenfolgen nicht überein. Daher ist unsere Annahme falsch und es existiert keine Lösung.
   b) $m = 3$ mit Indizes $(2, 1, 3)$

11. SL_Blatt

entscheidbar	unentscheidbar
das Erfüllbarkeitsproblem der Aussagenlogik (SAT)	das Halteproblem
das Wortproblem der Boolschen Algebra	das Postsche Korrespondenzproblem
sei G eine KFG, ist $L(G) = \emptyset$?	sei Geine KFG über Σ, ist $L(G) = \Sigma^*$?
$\ldots$	$\ldots$

Turingmaschine (TM)

deterministisch, einbändige Turingmaschine (TM)



- Eine TM verwendet ein einseitig unendliches Band als Speicher
- Zu Beginn der Berechnung steht die Eingabe auf dem Band
- Der Speicher wird mit einem Lese/Schreibkopf gelesen oder beschrieben
- Das Verhalten der TM wird durch die endliche Kontrolle Q kontrolliert

eine deterministische, einbändige Turingmaschine (TM) M ist ein 9-Tupel

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

sodass

1. Q eine endliche Menge von Zuständen,
2. Σ eine endliche Menge von Eingabesymbolen,
3. Γ eine endliche Menge von Bandsymbolen, mit $\Sigma \subseteq \Gamma$,
4. $\vdash \in \Gamma \setminus \Sigma$, der linke Endmarker,
5. $\sqcup \in \Gamma \setminus \Sigma$, das Blanksymbol,
6. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ die Übergangsfunktion,
7. $s \in Q$, der Startzustand,
8. $t \in Q$, der akzeptierende Zustand und
9. $r \in Q$, der verwerfende Zustand mit $t \neq r$.

6) das Kreuzprodukt ist wie ein Komma

Übergangsfunktion

die Gleichung $\delta(p, a) = (q, b, d)$ bedeutet: wenn die TM M im Zustand p das Symbol a liest, dann

1. M ersetzt a durch b auf dem Band
2. der Lese/Schreibkopf bewegt sich einen Schritt in die Richtung d
3. M wechselt in den Zustand q

Definition (Zusatzbedingungen)

- Der linke Endmarker darf nicht überschreiben werden

  $\forall p \in Q, \exists q \in Q \quad \delta(p, \vdash) = (q, \vdash, R)$

- Wenn die TM akzeptiert/verwirft, bleibt die TM in diesem Zustand

$$\forall\, b \in \Gamma,\ \exists\, c, c' \in \Gamma \text{ und } d, d' \in \{\mathrm{L}, \mathrm{R}\}: \quad \begin{aligned} \delta(t, b) &= (t, c, d) \\ \delta(r, b) &= (r, c', d') \end{aligned}$$

Sollte einmal ein verwerfender Zustand eingelesen werden, dann bleibt die TM im verwerfenden Zustand

Beispiel einer Turing Maschine (komplex):

sei $M = (\{s, t, r, q_1, q_2, q_3\}, \{0, 1\}, \{\vdash, \sqcup, 0, 1, \mathrm{X}, \mathrm{Y}\}, \vdash, \sqcup, \delta, s, t, r)$ mit δ wie folgt

$p \in Q$	$a \in \Gamma$	$\delta(p, a)$	$p \in Q$	$a \in \Gamma$	$\delta(p, a)$
s	$\vdash$	$(s, \vdash, \mathrm{R})$	q_2	$\vdash$	$(r, \vdash, \mathrm{R})$
s	$\sqcup$	$(r, \sqcup, \mathrm{R})$	q_2	$\sqcup$	$(r, \sqcup, \mathrm{R})$
s	0	$(q_1, \mathrm{X}, \mathrm{R})$	q_2	0	$(q_2, 0, \mathrm{L})$
s	1	$(r, 1, \mathrm{R})$	q_2	1	$(r, 1, \mathrm{R})$
s	X	$(r, \mathrm{X}, \mathrm{R})$	q_2	X	$(s, \mathrm{X}, \mathrm{R})$
s	Y	$(q_3, \mathrm{Y}, \mathrm{R})$	q_2	Y	$(q_2, \mathrm{Y}, \mathrm{L})$
q_1	$\vdash$	$(r, \vdash, \mathrm{R})$	q_3	$\vdash$	$(r, \vdash, \mathrm{R})$
q_1	$\sqcup$	$(r, \vdash, \mathrm{R})$	q_3	$\sqcup$	$(t, \sqcup, \mathrm{R})$
q_1	0	$(q_1, 0, \mathrm{R})$	q_3	0	$(r, 0, \mathrm{R})$
q_1	1	$(q_2, \mathrm{Y}, \mathrm{L})$	q_3	1	$(r, 1, \mathrm{R})$
q_1	X	$(r, \mathrm{X}, \mathrm{R})$	q_3	X	$(r, \mathrm{X}, \mathrm{R})$
q_1	Y	$(q_1, \mathrm{Y}, \mathrm{R})$	q_3	Y	$(q_3, \mathrm{Y}, \mathrm{R})$
t	$*$	$(t, *, \mathrm{R})$	r	$*$	$(r, *, \mathrm{R})$

Wir betrachten die Schrittfunktion für eine akzeptierende Berechnung von M bei Eingabe 0011:

$$\begin{aligned}(s, \vdash 0011\sqcup^\infty, 0) &\xrightarrow[M]{1} (s, \vdash 0011\sqcup^\infty, 1) \xrightarrow[M]{1} (q_1, \vdash X011\sqcup^\infty, 2)\\ &\xrightarrow[M]{1} (q_1, \vdash X011\sqcup^\infty, 3) \xrightarrow[M]{1} (q_2, \vdash X0Y1\sqcup^\infty, 2)\\ &\xrightarrow[M]{1} (q_2, \vdash X0Y1\sqcup^\infty, 1) \xrightarrow[M]{1} (s, \vdash X0Y1\sqcup^\infty, 2)\\ &\xrightarrow[M]{1} (q_1, \vdash XXY1\sqcup^\infty, 3) \xrightarrow[M]{1} (q_1, \vdash XXY1\sqcup^\infty, 4)\\ &\xrightarrow[M]{1} (q_2, \vdash XXYY\sqcup^\infty, 3) \xrightarrow[M]{1} (q_2, \vdash XXYY\sqcup^\infty, 2)\\ &\xrightarrow[M]{1} (s, \vdash XXYY\sqcup^\infty, 3) \xrightarrow[M]{1} (q_3, \vdash XXYY\sqcup^\infty, 4)\\ &\xrightarrow[M]{1} (q_3, \vdash XXYY\sqcup^\infty, 5) \xrightarrow[M]{1} (t, \vdash XXYY \sqcup \sqcup^\infty, 6)\end{aligned}$$

Registermaschinnen

Eine Registermaschine (RM) R ist ein Paar $R = ((x_i)_{1\leqslant i\leqslant n}, P)$ sodass

1. $(x_i)_{1\leqslant i\leqslant n}$ eine Sequenz von n Registern x_i, die natürliche Zahlen beinhalten
2. P ein Programm

Programme sind endliche Folgen von Befehlen und sind induktiv definiert:

1. folgende Instruktionen sind Befehle und Programme für Register x_i:
   a. $x_i := x_i + 1 \; und \; x_i := x_i - 1$
2. wenn P_1, P_2 Programme sind, dann ist $P_1; P_2$ ein Programm
3. wenn P_1 ein Programm und x_i ein Register, dann ist $while \; x_i \neq 0 \; do \; P_1 \; end$ ein Befehl und auch ein Programm

1. Zu Beginn der Berechnung steht die Eingabe (als natürliche Zahlen) in den Registern
2. Die Befehle
   - $x_i := x_i + 1$
   - $x_i := x_i - 1$

   bedeuten, dass der Inhalt des Register x_i entweder um 1 erhöht oder vermindert wird
3. $P_1; P_2$ bedeutet, dass zunächst das Programm P_1 und dann das Programm P_2 ausgeführt wird
4. Der Befehl (und das Programm)

   $$\text{while } x_i \neq 0 \text{ do } P_1 \text{ end}$$

   bedeutet, der Schleifenrumpf P_1 wird ausgeführt, bis die Bedingung $x_i \neq 0$ falsch ist

Sei $R = ((x_i)_{1 \leqslant i \leqslant 5}, P)$ eine RM mit dem folgenden Programm:

Zuweisung $x_i := x_j$

```
while x_i ≠ 0 do
  x_i := x_i − 1
end;
while x_k ≠ 0 do
  x_k := x_k − 1
end
```

```
while x_j ≠ 0 do
  x_i := x_i + 1;
  x_j := x_j − 1;
  x_k := x_k + 1
end;
while x_k ≠ 0 do
  x_j := x_j + 1;
  x_k := x_k − 1
end
```

Multiplikation

```
x_3 := 0;
while x_1 ≠ 0 do
  x_1 := x_1 − 1;
  x_4 := x_2;
  while x_2 ≠ 0 do
    x_2 := x_2 − 1;
    x_3 := x_3 + 1
  end;
  x_2 := x_4
end
```

Bei Eingabe $(m, n, 0, 0, 0)$ berechnet R $(0, n, m \times n, n, 0)$

Komplexitätstheorie

Die Komplexitätstheorie analysiert Algorithmen und Probleme:

Welche Ressourcen benötigt ein bestimmter Algorithmus oder ein Problem?

Ressourcen:

- Speicherplatz
- Rechenzeit
- ...

Was ist die Komplexität eines Algorithmus:

Algorithmus von Quine: 2^{c*n} (n ist die maximale Länge der Eingabe)

Was ist die Komplexität eines Problems:

SAT ist in NP

Laufzeitkomplexität

Definition

sei M eine totale TM

- die Laufzeitkomplexität von M ist Funktion $T : \mathbb{N} \to \mathbb{N}$, wobei T wie folgt definiert
$$T(n) := \max\{m \mid M \text{ hält bei Eingabe } x, |x| = n, \text{ nach } m \text{ Schritten}\}$$
- $T(n)$ bezeichnet die Laufzeit von M, wenn n die Länge der Eingabe
- M heißt T-Zeit-Turingmaschine

Definition

sei $T : \mathbb{N} \to \mathbb{N}$ eine numerische Funktion
$$\mathrm{DTIME}(T) := \{\mathrm{L}(M) \mid M \text{ ist eine mehrbändige TM mit Laufzeit (ungefähr) in } T\}$$

NB: Formal gilt $\exists c \in \mathbb{R}^+ \; \exists m \; \forall n \geqslant m$ Laufzeit von M bei Eingabe $x \leqslant c \cdot T(n)$, wobei $|x| = n$.

Die Klasse P und NP

Definition

$$\mathsf{P} := \bigcup_{k \geqslant 1} \mathrm{DTIME}(n^k)$$

Beispiel

betrachte SAT als Sprache:

$$\mathrm{SAT} = \{F \mid F \text{ Formel mit erfüllbarer Belegung v}\}$$

es gilt $\mathrm{SAT} \in \mathrm{DTIME}(2^n)$, aber es ist nicht bekannt ob $\mathrm{SAT} \in \mathsf{P}$

Definition

- ein Verifikator einer Sprache $L \subseteq \Sigma^*$, ist ein Algorithmus V sodass
$$L = \{x \in \Sigma^* \mid \exists\, c, \text{ sodass } V \text{ akzeptiert Eingabe } (x, c)\}$$
- ein polytime Verifikator ist ein Verifikator mit (ungefährer) Laufzeit n^k wobei $|x| = n$
- Wort c wird Zertifikat genannt

Definition

NP ist die Klasse der Sprachen, die einen polytime Verifikator haben

Beispiel

- Es gilt $\mathrm{SAT} \in \mathrm{NP}$.
- Als Zertifikat wählen wir die (erfüllende) Belegung v für F. Für jede Belegung v kann leicht (in polynomieller Zeit) nachgewiesen werden, ob $\mathrm{v}(F) = \mathrm{T}$.

Reduktion (in polynomieller Zeit)

Definition

1. $\exists$ k-Band TM M mit Eingabealphabet Σ
2. M läuft in polynomieller Zeit
3. bei Eingabe $x \in \Sigma^*$, schreibt M, $R(x)$ auf das (erste) Band

dann heißt $R : \Sigma^* \to \Delta^*$ in polynomieller Zeit berechenbar

Definition

1. $\exists\, R : \Sigma^* \to \Delta^*$
2. R berechenbar in polynomieller Zeit
3. für $L \subseteq \Sigma^*$, $M \subseteq \Delta^*$ gilt $x \in L \Leftrightarrow R(x) \in M$

dann ist L in polynomieller Zeit auf M reduzierbar; kurz: $L \leqslant^{\mathrm{p}} M$

Beispiel (Wiederholung)

Seien

$$L = \{x \in \{a,b\}^* \mid |x| \text{ ist gerade}\}$$
$$M = \{x \in \{a,b\}^* \mid x \text{ ist ein Palindrom gerader Länge}\}$$

dann gilt $L \leqslant^p M$

Polynomielle Reduktion

Wir geben eine polynomiell berechenbare Abbildung $R\colon \{a,b\}^* \to \{a,b\}^*$ an, sodass $x \in L \Leftrightarrow R(x) \in M$:

- definiere R', sodass $R'(a) := a$ und $R'(b) := a$
- definiere R als Erweiterung von R' auf Wörter
- R ist eine Stringfunktion, die ein Wort aus $\{a,b\}^n$ in das Wort a^n umwandelt
- Genau dann wenn n gerade ist, ist a^n ein Palindrom gerader Länge

Definition

1. $\mathcal{C}$ eine beliebige Komplexitätsklasse
2. L eine Sprache über Σ
3. $\forall$ Sprachen $M \in \mathcal{C}$ gilt: $M \leqslant^p L$

dann ist L $\leqslant^p$-hart für $\mathcal{C}$

Beispiel

SAT ist $\leqslant^p$-hart für NP, dh. jedes Problem in NP ist auf SAT reduzierbar und außerdem ist SAT $\in$ NP

Definition

für eine Sprache L, sei

1. L $\leqslant^p$-hart für $\mathcal{C}$ und
2. $L \in \mathcal{C}$

dann ist L $\leqslant^p$-vollständig für $\mathcal{C}$ oder (kurz) $\mathcal{C}$-vollständig

Verifikation nach Hoare

Prädikatenlogik (informell)

- die Ausdruckskraft ist hier stärker als bei der Aussagenlogik
- Quantoren sind die wichtigen Prädikatensymbole
- Prädikatensymbole erlauben es uns, über Elemente einer Menge Aussagen zu treffen

Sprache einer Prädikatenlogik

Eine Prädikatenlogik durch eine Sprache beschrieben, diese Sprache enthält:

1. Funktionssymbole und Prädikatensymbole; Variablen
2. $\underbrace{=}_{\text{Gleichheit}}, \underbrace{\neg, \wedge, \vee, \to,}_{\text{Junktoren}} \underbrace{\forall, \exists}_{\text{Quantoren}}$

Beispiel

- Sei 7 eine Konstante und ist_prim ein Prädikatensymbol
- Wir schreiben ist_prim(7), um auszudrücken, dass 7 eine Primzahl

Ein Ausdruck der mit Hilfe von Variablen und Funktionssymbolen gebildet wird, heißt **Term**

1. Sei P ein Prädikatssymbol
2. Seien $t_1, \ldots, t_n$ Terme

Dann nehmen wir die Ausdrücke $P(t_1, \ldots, t_n)$ und $t_1 = t_2$ Atome oder atomare Formel.

Zusicherungen

1. Atome sind Zusicherungen
2. Wenn A und B Zusicherungen sind, dann sind auch die folgenden Ausdrücke, Zusicherungen:
    a. $\neg A \ \ (A \wedge B) \ \ (A \vee B) \ \ (A \rightarrow B)$

Zusicherungen werden Formeln genannt.

Interpretationen I werden verwendet, um den Ausdrücken der Prädikatenlogik eine Bedeutung zu geben.

Beispiel

- Wir betrachten die Konstante 7 und das Prädikat ist_prim
- Interpretation $\mathcal{I}$ legt fest, dass 7 als die Zahl sieben zu verstehen ist
- $\mathcal{I}$ legt fest, dass das Atom ist_prim(n) genau dann wahr ist, wenn n eine Primzahl

Beobachtung

1. Mit Hilfe von Interpretationen wird der Wahrheitsgehalt von Atomen bestimmt
2. Ist die Wahrheit von Atomen in $\mathcal{I}$ definiert, wird die Wahrheit einer beliebigen Formel durch die Bedeutung der Junktoren bestimmt

Definition

Sei $\mathcal{I}$ eine Interpretation und F eine Formel, wir schreiben $\mathcal{I} \models F$, wenn die Formel F in der Interpretation $\mathcal{I}$ wahr ist

Beispiel

Wenn x die Primzahl 7 ist, dann gilt $\mathcal{I} \models \text{ist\_prim}(x) \wedge x = 7$

Definition

Die Konsequenzrelation $A \models B$ gilt, gdw. für alle Interpretationen $\mathcal{I}$:

$$\mathcal{I} \models A \text{ impliziert } \mathcal{I} \models B$$

Beispiel

Seien $x_1 > 4$ und $x_1 + 1 > 5$ Atome, es gilt:

$$x_1 > 4 \models x_1 + 1 > 5$$

Hoare-Tripel

- Sei P ein while-Programm (ein Programm einer Registermaschine)
- Seien Q und R Zusicherungen
- Ein Hoare-Tripel ist wie folgt definiert:
    - $\{Q\}\ P\ \{R\}$
- Q wird die **Vorbedingung**
- R wird die **Nachbedingung**

Beispiel

Seien $x_1 > 4$, $x_1 > 5$ Zusicherungen und $x_1 := x_1 + 1$ ein Programm,
dann ist $\{x_1 > 4\}\ x_1 := x_1 + 1\ \{x_1 > 5\}$ ein Hoare-Tripel

Wann ist ein Hoare Tripel wahr?

1. wenn Q **vor** der Ausführung von P gilt
2. wenn R **nach** der Ausführung von P gilt
3. unter der Voraussetzung, dass P terminiert

partiell korrekt:

wenn das Programm P nicht unbedingt terminiert

total korrekt:

wenn das Programm P korrekt ist und auch terminiert

Beispiel

Die folgenden Hoare-Tripel

$$\{x_1 > 4\}\ x_1 := x_1 + 1\ \{x_1 > 5\} \qquad \{x_2 = 0\}\ x_2 := x_2 - 1\ \{x_2 = 0\}$$

sind wahr und die jeweiligen Programme total korrekt

Regeln des Hoare Kalkül:

Definition

Die Regeln des Hoare-Kalkül sind wie folgt definiert:

$$[z]\ \frac{}{\{Q\{x \mapsto t\}\}\ x := t\ \{Q\}} \qquad [a]\ \frac{\{Q'\}\ P\ \{R'\}}{\{Q\}\ P\ \{R\}}\ Q \models Q', R' \models R$$

$$[s]\ \frac{\{Q\}\ P_1\ \{R\} \quad \{R\}\ P_2\ \{S\}}{\{Q\}\ P_1;P_2\ \{S\}} \qquad [w]\ \frac{\{I \land B\}\ P\ \{I\}}{\{I\}\ \text{while } B \text{ do } P \text{ end}\ \{I \land \neg B\}}$$

Ist ein Hoare-Tripel in diesem Kalkül ableitbar, dann ist es wahr

Regel $|z|$: ist die letzte Regel

Regel $|a|$: schwächt die Aussagen aus, wichtig ist zu nennen was aus was folgt

Regel $|s|$: trennt das Program in zwei Unterprogramme; WICHTIG NUR ZWEI PROGRAMME ZU TRENNEN

Regel $|w|$: löst die Bedingung der while-Schleife auf und verknüpft diese mit der Interferenzregel

Wir betrachten das folgende einfache while-Programm P:

while $x_i \neq 0$ do
  $x_i := x_i - 1$
end

und zeigen $\{x_i \geqslant 0\}\ P\ \{x_i = 0\}$

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\{x_i - 1 \geqslant 0\}\ x_i := x_i - 1\ \{x_i \geqslant 0\}}\ [z]}{\{x_i \geqslant 0 \land x_i \neq 0\}\ x_i := x_i - 1\ \{x_i \geqslant 0\}}\ [a]}{\{x_i \geqslant 0\}\ P\ \{x_i \geqslant 0 \land x_i = 0\}}\ [w]}{\{x_i \geqslant 0\}\ P\ \{x_i = 0\}}\ [a]$$

wir verwenden:

1. $x_i \geqslant 0 \land x_i = 0 \models x_i = 0$
2. die Schleifeninvariante $x_i \geqslant 0$
3. $x_i \geqslant 0 \land x_i \neq 0 \models x_i - 1 \geqslant 0$

Beispiele:

$$
\dfrac{\dfrac{\dfrac{}{\{x_1+1=1\}\; x_1 := x_1+1\; \{x_1=1\}}\,[\text{z}]}{\{x_1=0\}\; x_1 := x_1+1\; \{x_1=1\}}\,[\text{a}]^1 \qquad \dfrac{\dfrac{\dfrac{}{\{x_1-1=0\}\; x_1 := x_1-1\; \{x_1=0\}}\,[\text{z}]}{\{x_1=1\}\; x_1 := x_1-1\; \{x_1=0\}}\,[\text{a}]^2 \quad \dfrac{\dfrac{}{\{\mathsf{odd}(x_1+1)\}\; x_1 := x_1+1\; \{\mathsf{odd}(x_1)\}}\,[\text{z}]}{\{x_1=0\}\; x_1 := x_1+1\; \{\mathsf{odd}(x_1)\}}\,[\text{a}]^3}{\{x_1=1\}\; x_1 := x_1-1; x_1 := x_1+1\; \{\mathsf{odd}(x_1)\}}\,[\text{s}]}{\{x_1=0\}\; x_1 := x_1+1; x_1 := x_1-1; x_1 := x_1+1\; \{\mathsf{odd}(x_1)\}}\,[\text{s}]
$$

[1] mit $(x_1 = 0) \models (x_1 + 1 = 1)$
[2] mit $(x_1 = 1) \models (x_1 - 1 = 0)$
[3] mit $(x_1 = 0) \models (\mathsf{odd}(x_1 + 1))$

$$
\dfrac{\dfrac{\dfrac{}{\{x_1+1=1\}\; x_1 := x_1+1\; \{x_1=1\}}\,[\text{z}]}{\{x_1=0\}\; x_1 := x_1+1\; \{x_1=1\}}\,[\text{a}]^1 \qquad \Phi}{\{x_1=0\}\; x_1 := x_1+1; x_1 := x_1+1; x_1 := x_1+1\; \{\mathsf{ist\_prim}(x_1)\}}\,[\text{s}]
$$

$$
\dfrac{\dfrac{\dfrac{}{\{x_1+1=2\}\; x_1 := x_1+1\; \{x_1=2\}}\,[\text{z}]}{\{x_1=1\}\; x_1 := x_1+1\; \{x_1=2\}}\,[\text{a}]^2 \qquad \Psi}{\Phi := \{x_1=1\}\; x_1 := x_1+1; x_1 := x_1+1\; \{\mathsf{ist\_prim}(x_1)\}}\,[\text{s}]
$$

$$
\dfrac{\dfrac{}{\{\mathsf{ist\_prim}(x_1+1)\}\; x_1 := x_1+1\; \{\mathsf{ist\_prim}(x_1)\}}\,[\text{z}]}{\Psi := \{x_1=2\}\; x_1 := x_1+1\; \{\mathsf{ist\_prim}(x_1)\}}\,[\text{a}]^3
$$

[1] mit $(x_1 = 0) \models (x_1 + 1 = 1)$
[2] mit $(x_1 = 1) \models (x_1 + 1 = 2)$
[3] mit $(x_1 = 2) \models (\mathsf{ist\_prim}(x_1 + 1))$