

Was bisher geschah

Wiederholung Berechnungsmodell DTM (deterministische Turing-Maschine)

- ▶ DTM $M = (X, Q, \Gamma, \delta, q_0, \square)$
- ▶ Konfigurationen
- ▶ Konfigurationsfolgen (Berechnungen)
- ▶ Akzeptanz durch Halt
- ▶ Nebenwirkung: Änderung des Bandinhaltes
- ▶ Mehrband-TM, Simulation durch Einband-TM
- ▶ Berechnung von Funktionen $X^* \rightarrow \Gamma^*$ durch DTM
- ▶ Berechnung von Funktionen $\mathbb{N} \rightarrow \mathbb{N}$ durch DTM

Darstellung von Daten

- ▶ DTM können Zeichenketten akzeptieren und manipulieren (Nebenwirkung der Ausführung)
- ▶ Zeichenketten über (endlichem) Alphabet A können als Zahlen zur Basis $|A|$ oder $|A| + 1$ interpretiert werden.

Beispiele für Bedeutungen von Zeichenketten:

- ▶ Wörter aus Sprachen
- ▶ Zahlen
- ▶ Darstellung strukturierter Daten, z.B.
 - ▶ Tupel,
 - ▶ Listen,
 - ▶ Bäume,
 - ▶ Graphen,
 - ▶ NFA, TM, ...
 - ▶ Konfigurationen, Konfigurationsfolgen
 - ▶ Programmtexte,
 - ▶ ...

Darstellung von Zahlen als Zeichenketten

- ▶ natürliche Zahlen: Unär-, Binär-, ...-Darstellung
in Unär-Darstellung häufig sinnvoll : $\forall n \in \mathbb{N} : f(n) = 1^{n+1}$
- ▶ ganze, gebrochene Zahlen: Paare natürlicher Zahlen
mit um Trennzeichen erweitertem Alphabet,
z.B. $2/3$ codiert durch Paar $(2, 3)$ in Unärdarstellung mit
Trennzeichen 0 : 11101111
- ▶ übliche Approximationen reeller Zahlen durch
Maschinenzahlen

Darstellung von Zeichenketten als natürliche Zahlen

Darstellung des Alphabets (endliche Menge):

1. (beliebige) Reihenfolge der Symbole in A festlegen:

$$A = \{a_1, \dots, a_{|A|}\}$$

2. Nummerierung der Symbole:

$$\text{Zuordnung } \forall i \in \{1, \dots, |A|\} : c(a_i) = i \in \mathbb{N}$$

3. zur Weiterverarbeitung: Unärdarstellung $c' : A \rightarrow 1^*$ mit

$$\forall i \in \{1, \dots, |A|\} : c'(a_i) = 1^i$$

Beispiel: $A = \{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\}$,

Nummerierung $c : A \rightarrow \mathbb{N}$ mit $c(\clubsuit) = 1, c(\spadesuit) = 2, c(\heartsuit) = 3, c(\diamondsuit) = 4$
und $c' : A \rightarrow 1^+$ mit $c'(\clubsuit) = 1, c'(\spadesuit) = 11, c'(\heartsuit) = 111, c'(\diamondsuit) = 1111$

Darstellung von Wörtern $w \in A^*$ über endlichem Alphabet A :

1. Nummerierung und Unärdarstellung der Alphabet-Symbole (s.o.),

2. Verkettung mit 0 als Trennsymbol

$$c'(w) = c'(w_1 \dots w_{|w|}) = c'(w_1)0c'(w_2)0 \dots 0c'(w_{|w|})$$

3. Durch Interpretation der Zeichenkette $c'(w) \in \{0, 1\}^*$ als Binärzahl wird w die eindeutige Zahl $c''(w) \in \mathbb{N}$ zugeordnet.

Beispiel: $c'(\clubsuit\heartsuit\spadesuit) = 10111011, c''(\clubsuit\heartsuit\spadesuit) = 187$

Darstellung strukturierter Daten

- ▶ direkte Darstellung $c : D \rightarrow A^*$ der einfachen Daten $d \in D$
(oder Numerierung in Unärdarstellung analog Alphabet)
- ▶ Codierung von Paaren, Tupeln, Listen $(x_1, \dots, x_n)$ als Wörter
z.B. $(5, \heartsuit, 0, 1) \mapsto 111111011101011$
- ▶ Schachtelungen z.B. durch neue oder mehrfache Trennzeichen
z.B. $(5, (\heartsuit, 0), 1) \mapsto 111111 \bullet 11101 \bullet 11$
oder $(5, (\heartsuit, 0), 1) \mapsto 11111100111010011$

Beispiel: Darstellung von Graphen

(endliche gerichtete) Graphen $G = (V, E)$ mit $E \subseteq V \times V$

Idee:

Betrachte V als Alphabet und E als Folgen der Länge 2 über V

- ▶ (beliebige) Reihenfolge der Knoten festlegen $V = \{v_1, \dots, v_{|V|}\}$
- ▶ Numerieren der Knoten, z.B. unär: $c_V : V \rightarrow 1^+$ mit
 $\forall i \in \{1, \dots, |V|\} : c_V(v_i) = 1^{i+1}$
- ▶ Darstellung der Kanten: $c_E(u, v) = c_V(u)0c_V(v)$
- ▶ $c(G) = c_V(v_1)0 \cdots 0c_V(v_{|V|})000c_E(u_1, v_1)00 \cdots 00c_E(u_{|E|}, v_{|E|})$

oder

- ▶ Darstellung jedes Knotens $v \in V$ als Paar
($c_V(v)$, Darstellung der Liste der Namen aller Nachbarn)
(ggf. als Tupel mit weiteren Daten, z.B. Schlüssel)
- ▶ $c(G)$ = Darstellung von G Graph als Codierung der Liste aller Knoten (Adjazenzliste)

oder ...

Gödelisierung

(Kurt Gödel 1906 - 1978)

- ▶ Die bisher vorgestellten Codierungen c ordnen jedem Element x der darzustellenden Menge (Graphen, TM, Konfigurationen, ...) ein **eindeutiges** Wort $c(x) \in \{0, 1\}^*$ zu.
 - ▶ Interpretation der Zeichenkette $c(x) \in \{0, 1\}^*$ als Binärzahl ordnet jedem Element x der darzustellenden Menge eine **eindeutige** natürliche Zahl $c'(x) \in \mathbb{N}$ zu.
1. Beide Funktionen c und c' sind injektiv und Turing-berechenbar.
 2. Es lässt sich algorithmisch (z.B. durch TM) feststellen, ob ein Wort $w \in \{0, 1\}^*$ (eine Zahl $n \in \mathbb{N}$) die Darstellung eines Elementes ist. (ÜA: Wie?)
 3. Die Umkehrfunktionen $c^{-1} : c(X) \rightarrow X$ und $c'^{-1} : c'(X) \rightarrow X$ sind injektiv.
Aus jeder korrekten Darstellung lässt sich das Element x eindeutig algorithmisch (z.B. durch TM) berechnen.

Solche Funktionen c (und c') heißen **Gödelisierungen**.

Für jede Gödelisierung c heißt $c(x)$ **Gödelnummer** des Elementes x .

Darstellung von TM

Definition von TM $M = (X, Q, \Gamma, \delta, q_0, \square)$

Codierung als endliches Wort über dem endlichen Alphabet

$X' = X \cup \Gamma \cup Q \cup \{L, R, N\} \cup \text{Klammern} \cup \text{Trennzeichen möglich}$

Die Menge aller solcher Darstellungen von TM ist eine Sprache über X' .

Darstellung der TM als $\{0, 1\}$ -Folge:

- ▶ Alphabet, Zustände, Richtungen in Unärcodierung (aus 1^+)
- ▶ Trennzeichen als Folgen aus 0^+

Die Menge aller solcher Darstellungen von TM ist eine Sprache über $\{0, 1\}$.

ÜA: Chomsky-Typ der Sprache aller $\{0, 1\}$ -Folgen, die in dieser Darstellung korrekte Codierungen von TM sind

Beispiel: Darstellung von TM

Darstellung von TM als Wort aus $\{0, 1\}^*$

Beispiel: TM $M = (X, Q, \Gamma, \delta, q_0, \square)$ in $\{0, 1\}^*$ mit

- ▶ $X = \{1\}$, $\Gamma = \{1, \square\}$ mit $c(1) = 1$, $c(\square) = 11$
- ▶ $Q = \{q_0, q_1\}$ mit Startzustand q_0 : $c(q_0) = 1$, $c(q_1) = 11$
- ▶ $B = \{L, R, N\}$ mit $c(L) = 1$, $c(R) = 11$, $c(N) = 111$
- ▶ $\delta = \{(\square, q_0, 1, q_1, R), (\square, q_1, 1, q_0, L), (1, q_0, 1, q_1, L)\}$

für jeden Übergang $u = (a, p, b, q, r) \in \delta$:

Darstellung : $c(u) = c(a)0c(p)0c(b)0c(q)0c(r)$

$$c(\square, q_0, 1, q_1, R) = \underbrace{11}_{\square} 0 \underbrace{1}_{q_0} 0 \underbrace{1}_1 0 \underbrace{11}_{q_1} 0 \underbrace{11}_R$$

$$c(\square, q_1, 1, q_0, L) = 11011010101$$

$$c(1, q_0, 1, q_1, L) = 1010101101$$

Darstellung der TM M (δ genügt):

$$\begin{aligned} c(M) &= c(\square, q_0, 1, q_1, R)00c(\square, q_1, 1, q_0, L)00c(1, q_0, 1, q_1, L) \\ &= 110101011011001101101010101001010101101 \end{aligned}$$

Darstellung von TM-Konfigurationen

Konfigurationen einer TM $M = (X, Q, \Gamma, \delta, q_0, \square)$:

$upv \in \Gamma^* \times Q \times \Gamma^*$

- ▶ Darstellung von u und v als $\{0, 1\}$ -Folgen (wie bisher)
- ▶ Darstellung von p als Unärzahl
- ▶ Markierung der Kopfposition
(Position der als Zustand zu interpretierenden Zeichenkette)
z.B. durch doppeltes Trennzeichen nach dem Zustand

Beispiel: TM von voriger Folie

Startkonfiguration $\underbrace{\square}_u q_0 \underbrace{11}_v \mapsto \underbrace{11}_{c(u)} 0 \underbrace{1}_{c(q_0)} 00 \underbrace{101}_{c(v)}$

Universelle (programmierbare) TM

Definition einer DTM $M = (X, Q, \Gamma, \delta, q_0, \square)$ ist eine
endliche Beschreibung zur
Berechnung einer partiellen Funktion $f : X^* \rightarrow \Gamma^*$

Eingabe: $w \in X^*$

Ausgabe: ▶ Halt und $f_M(w) \in \Gamma^*$ oder
 ▶ kein Halt

Es existieren universelle TM (programmierbare TM) U , die

- ▶ endliche Beschreibungen jeder TM M interpretieren und damit
- ▶ die Berechnungen jeder TM M simulieren
(den durch die TM definierten Algorithmus ausführen) kann
insbesondere gilt $\forall$ DTM $M \forall n \in \mathbb{N} : f_U(c(M), n) = f_M(n)$

endliche Beschreibung einer TM ist Programm zur
Berechnung einer Funktion / Ausführung eines Algorithmus
(durch eine universelle TM)

Darstellung von Problemen als Sprachen

Aufgaben (Probleme): Zuordnung zwischen

- ▶ Eingaben
- ▶ passenden Ausgaben

(oft strukturierte Daten)

Aufgabe: Abbildung = Menge von Paaren (Eingabe, Ausgabe)
(übliche Codierungen)

Spezialfall Entscheidungsprobleme: Ausgabe $\in \{0, 1\}$

- ▶ definiert eine Eigenschaft auf der Menge aller Eingaben
- ▶ repräsentiert durch Urbild der 1
- ▶ charakteristische Funktion der Menge der Eingaben mit dieser Eigenschaft

Aufgaben und Instanzen – Beispiele

- ▶ Enthaltensein in einer Folge (von Elementen des Typs E)

Aufgabe: $F = \{((x_1, \dots, x_n), y) \in E^* \times E \mid$
 $\exists k \in \{1, \dots, n\} : x_k = y\}$

Menge aller Paare $((x_1, \dots, x_n), y)$ mit
 $y \in \{x_1, \dots, x_n\}$

Instanz: $((1, 3, 4, 6), 4)$ (Gilt $((1, 3, 4, 6), 4) \in F$?)

Ja, $((1, 3, 4, 6), 4) \in F$, weil $4 \in \{1, 3, 4, 6\}$

Lösungsverfahren: (beliebiges) Suchverfahren in Folgen

- ▶ SAT (Erfüllbarkeitsproblem der Aussagenlogik)

Aufgabe: $\text{SAT} = \{\varphi \in \text{AL} \mid \text{Mod}(\varphi) \neq \emptyset\}$

Menge aller erfüllbaren aussagenlogischen Formeln

Instanz: $\neg a \wedge (a \vee b \vee c) \wedge \neg c \wedge \neg b$

(Gilt $(\neg a \wedge (a \vee b \vee c) \wedge \neg c \wedge \neg b) \in \text{SAT}$?)

Nein, weil $\text{Mod}(\varphi) = \emptyset$

Lösungsverfahren (Entscheidungsverfahren):

- ▶ semantisch: Wahrheitstabelle
- ▶ syntaktisch: Kalküle, z.B. Resolution, äquivalente Umformungen von ... zu ...

Beispiel: Graphen-Probleme

- ▶ k -Färbung eines Graphen $G = (V, E)$:
Abbildung $f : V \rightarrow \{1, 2, \dots, k\}$
- ▶ Färbung f von $G = (V, E)$ ist konfliktfrei
gdw. $\forall xy \in E : f(x) \neq f(y)$

$$k\text{COL} = \{G \mid \exists f : f \text{ ist konfliktfreie } k\text{-Färbung von } G \}$$

Hamilton-Kreis in $G = (V, E)$:

Kreis in G durch alle Knoten in V

$\text{HC} = \{G \mid G \text{ ist ungerichteter Graph und } G \text{ enthält Hamilton-Kreis} \}$ (Hamiltonian Circuit)

$\text{DHC} = \{G \mid G \text{ ist gerichteter Graph und } G \text{ enthält gerichteten Kreis durch alle Knoten} \}$ (Directed HC)

Beispiel: TM-Probleme als Sprachen $\subset \{0, 1\}^*$

$$\{w \in \{0, 1\}^* \mid \exists \text{ TM } M \text{ mit } w = c(M)\}$$

Menge aller Binärwörter, die korrekte Codierungen von TM sind

$$\{w \in \{0, 1\}^* \mid \exists \text{ TM } M = (X, Q, \Gamma, \delta, q_0, \square) \text{ mit } w = c(M) \text{ und } Q = \{q_0\}\}$$

Menge aller (Codierungen von) TM mit genau einem Zustand

$$L = \{w \in \{0, 1\}^* \mid \exists \text{ TM } M \text{ mit } w = c(M) \text{ und } L_M = \emptyset\}$$

Menge aller TM, die keine Eingabe akzeptieren

$$H = \{w \in \{0, 1\}^* \mid \exists \text{ TM } M \text{ mit } w = c(M) \text{ und } w \in L_M\}$$

Menge aller TM, die bei Eingabe ihrer Codierung halten
(spezielles Halteproblem)

$$D = \{w \in \{0, 1\}^* \mid \exists \text{ TM } M \text{ mit } w = c(M) \text{ und } w \notin L_M\}$$

Menge aller TM, die bei Eingabe ihrer Codierung nicht halten
(Diagonalsprache)

ÜA: Warum gilt $D \neq \overline{H}$?

Codierung in natürlichen Zahlen

DTM akzeptieren / manipulieren Zeichenketten.

Die meisten Berechenbarkeitsmodelle (z.B. Programme, rekursive Funktionen) rechnen aber mit natürlichen Zahlen.

Das ist keine Einschränkung der Allgemeinheit, denn man kann jedes strukturierte Datum in eine einzige natürliche Zahl codieren.

Vorgehen wie in den vorangegangenen Beispielen:

1. Darstellung als Zeichenkette $c(d) \in \{0,1\}^*$
2. Interpretation von $c(d)$ als Binärzahl $c'(d) \in \mathbb{N}$

Bei Berechnungen mit Zahlen ist der Umweg über Zeichenketten oft zu aufwendig.

deshalb gesucht:

direkte Codierung und Decodierung strukturierter Daten in natürlichen Zahlen

Codierung von Zahlenpaaren

gesucht sind (TM-)berechenbare Funktionen

Konstruktor: $C : \mathbb{N}^2 \rightarrow \mathbb{N}$

Destruktoren: $P_1, P_2 : \mathbb{N} \rightarrow \mathbb{N}$

Testfunktion: $T : \mathbb{N} \rightarrow \{0, 1\}$

mit den folgenden Eigenschaften (Spezifikation):

$$\forall x_1, x_2 \in \mathbb{N} : P_1(C(x_1, x_2)) = x_1$$

$$\forall x_1, x_2 \in \mathbb{N} : P_2(C(x_1, x_2)) = x_2$$

$$\forall x \in \mathbb{N} : T(x) = 1 \iff \exists x_1, x_2 \in \mathbb{N} : x = C(x_1, x_2)$$

verschiedene Möglichkeiten, z.B.:

- ▶ $C(x_1, x_2) = (x_1 + x_2)(x_1 + x_2 + 1)/2 + x_1$
- ▶ $C(x_1, x_2) = 2^{x_1}(2x_2 + 1),$
- ▶ $C(x_1, x_2) = 2^{x_1} \cdot 3^{x_2}$

ÜA: jeweils $C(2, 3), C(3, 2), T(10), T(12), P_1(12), P_2(12),$
Algorithmen für T, P_i .

Codierung von Listen

Konstruktor $L : \mathbb{N}^* \rightarrow \mathbb{N}$

Destruktoren $D_i : \mathbb{N} \rightarrow \mathbb{N}$.

zwei (von vielen) Möglichkeiten:

- ▶ mittels einer Paar-Codierung C

IA: $L([]) = 0$

IS: $\forall a \in \mathbb{N} \forall w \in \mathbb{N}^* : L([a] \circ w) = C(a + 1, L(w))$

- ▶ direkte Kodierung als Produkt von Primzahlpotenzen

$$L([x_0, x_1, \dots, x_n]) = 2^{x_0+1} \cdot 3^{x_1+1} \cdot \dots \cdot p(n)^{x_n+1}.$$

ÜA: jeweils $L([])$, $L([5])$, $L([2, 3, 0])$, Algorithmus für D_i

ÜA: die Funktion $p : n \mapsto$ die n -te Primzahl, also

$p(0) = 2, p(1) = 3, p(2) = 5, \dots$ ist algorithmisch berechenbar.

Codierung von Bäumen

Motivation

- ▶ allgemein: Baum = Term in einer Signatur,
- ▶ Signatur: endlichen Menge von Funktionssymbolen mit zugeordneter Stelligkeit.
- ▶ Bsp: $\Sigma = \{(f, 2), (g, 1), (a, 0)\}$,
 $t = f(f(a, g(a)), a) \in \text{Term}(\Sigma)$.
Notation $\text{root}(t) = f$, $\text{args}(t) = [f(a, g(a)), a]$.
- ▶ wird z.B. benötigt, um (prädikatenlogische) Formeln als Zahlen zu kodieren.

Realisierung: $B(t) = C(\text{num}(\text{root}(t)), L([B(t_1), \dots, B(t_k)]))$
mit $\text{args}(t) = [t_1, \dots]$, Paar-Kodierung C ,
Listen-Kodierung L , sowie Symbol-Nummerierung num für Σ