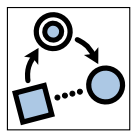
Komplexität und natürliche Sprache

Beschreibungskomplexität

Timm Lichte & Christian Wurm

Heinrich-Heine-Universität Düsseldorf

Sommersemester 2017, 21.06.2017



Outline

Outline

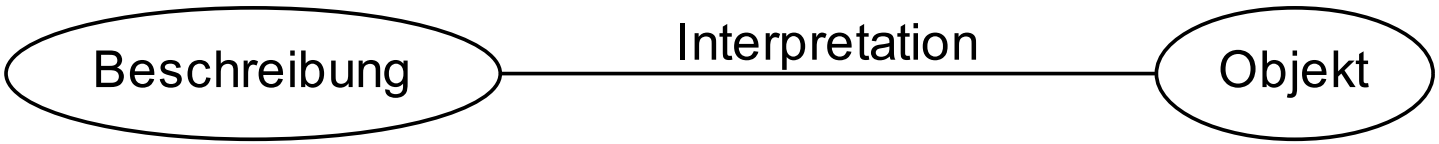
Rekapitulation: verschiedene Komplexitätsbegriffe

	Sprachklassen $K(\hat{L}_1) > K(\hat{L}_2)$	**Sprache** $K(L_1) > K(L_2)$	**Wort** $K(\overline{w_1}) > K(\overline{w_2})$
extensional	$\hat{L}_1 \supset \hat{L}_2$		
lernbar (Golds IiU)	$\hat{L}_1$ nicht lernbar, $\hat{L}_2$ lernbar		
algorithmisch	$K(\hat{\mathcal{M}}_{\hat{L}_1}) \supset K(\hat{\mathcal{M}}_{\hat{L}_2})$	$K(\mathcal{M}_{L_1}) > K(\mathcal{M}_{L_2})$	
deskriptiv			HEUTE
probabilistisch			NÄCHSTE WOCHE

- $K(X)$ ist die Komplexität von X
- $K(\hat{\mathcal{M}}_{\hat{L}})$ ist die niedrigste algorithmische Komplexitätsklasse (LIN-TIME,PTIME, …), zu der $\hat{L}$ gehört.
- $K(\mathcal{M}_L)$ ist der Speicher- oder Zeitbedarf des für L diesbezüglich optimalen Automaten $\mathcal{M}$ in $\mathcal{O}$-Notation.

Outline

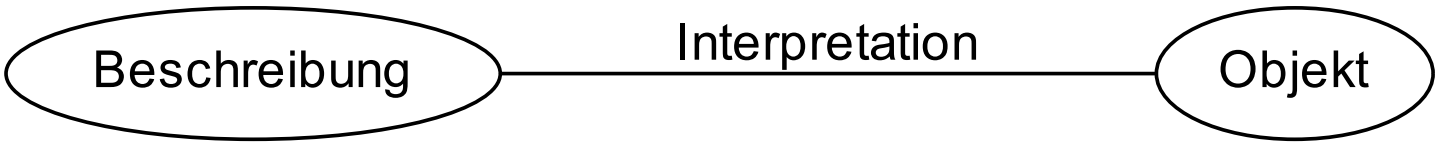
Beschreibung, Interpretation, Objekt



natürlich-sprachlicher Ausdruck
logische Formel
Programm, Automat, Symbol

String, Baum, Graph,
Mengen, …

Beschreibung, Interpretation, Objekt



natürlich-sprachlicher Ausdruck
logische Formel
Programm, Automat, Symbol

String, Baum, Graph,
Mengen, …

Annahmen:

- Beschreibungen sind **eindeutig**, d.h. ein Objekt ist mittels einer Beschreibung eindeutig in einer Menge von Objekten identifizierbar, d.h. die Interpretation ist eine Funktion von Beschreibungen nach Objekten.
- Im Folgenden nehmen wir der Einfachheit halber an, dass Beschreibungen und Objekte Strings sind.

Beschreibungssystem

Beschreibungssystem

Ein Beschreibungssystem ist ein Tupel $\langle \Sigma_s, \Sigma_P, S, P, f \rangle$ mit

- $S \subseteq \Sigma_S^+$, den Objekte,
- $P \subseteq \Sigma_P^+$, den Beschreibungen, und
- $f : P \longrightarrow S$, der surjektiven Interpretationsfunktion.

Beschreibungssystem $\langle \Sigma_s, \Sigma_P, S, P, f \rangle$ mit

- $\Sigma_S = \{1, 2, 3, 4, 5\}$
- $S = \{1, 2, 3, 4, 5\}$
- $\Sigma_P = \{a, \ldots, z\}$
- $P = \{eins, zwei, drei, vier, fuenf\}$
- $f = \{(eins, 1), (zwei, 2), (drei, 3), (vier, 4), (fuenf, 5)\}$

Im Weiteren kürzen wir $\langle \Sigma_s, \Sigma_P, S, P, f \rangle$ mit $\langle S, P, f \rangle$ ab.

Beschreibungskomplexität

Beschreibungskomplexität

Sei $\langle S, P, f \rangle$ ein Beschreibungssystem, dann ist die Beschreibungskomplexität K eines Objekts $s \in S$ bestimmt durch die Länge des kürzesten $p \in P$ mit $f(p) = s$, also

$$K_f(s) = \min_{f(p)=s} l(p)$$

Sei $l(fuenf) = 5$ und $l(eins) = l(zwei) = l(drei) = l(vier) = 4$.

Beschreibungskomplexität

Beschreibungskomplexität

Sei $\langle S, P, f \rangle$ ein Beschreibungssystem, dann ist die Beschreibungskomplexität K eines Objekts $s \in S$ bestimmt durch die Länge des kürzesten $p \in P$ mit $f(p) = s$, also

$$K_f(s) = \min_{f(p)=s} l(p)$$

Sei $l(fuenf) = 5$ und $l(eins) = l(zwei) = l(drei) = l(vier) = 4$.

Bei $\langle \{1, 2, 3, 4, 5\}, \{eins, zwei, drei, vier, fuenf\}, \{(eins, 1), (zwei, 2), (drei, 3), (vier, 4), (fuenf, 5)\} \rangle$ gilt dann

$$K(1) = K(2) = K(3) = K(4) < K(5)$$

Aber es gibt ja viele mögliche Beschreibungen für ein Objekt …

Beschreibungssysteme mit gleichbleibender Beschreibungskomplexität

Bei einem Beschreibungssystem $\langle S, P, f \rangle$ mit $|S| = n$ ist $P = \{0, 1\}^{log_2 n}$, d.h. $K(s) = log_2 n$ für jedes $s \in S$.

Bei $\langle \{1, 2, 3, 4, 5\}, \{001, 010, 011, 100, 101\}, \{(001, 1), (010, 2), (011, 3), (100, 4), (101, 5)\} \rangle$ gilt dann

$$K(1) = K(2) = K(3) = K(4) = K(5)$$

Beschreibungssysteme mit gleichbleibender Beschreibungskomplexität

Bei einem Beschreibungssystem $\langle S, P, f \rangle$ mit $|S| = n$ ist $P = \{0, 1\}^{log_2 n}$, d.h. $K(s) = log_2 n$ für jedes $s \in S$.

Bei $\langle \{1, 2, 3, 4, 5\}, \{001, 010, 011, 100, 101\}, \{(001, 1), (010, 2), (011, 3), (100, 4), (101, 5)\} \rangle$ gilt dann

$$K(1) = K(2) = K(3) = K(4) = K(5)$$

Einschränkung:

- nur für endliche Objektmengen
- keine inhärente Eigenschaft der Objekte

Beschreibungskomplexität: obere Grenze (upper bound)

Eine natürliche obere Grenze der Beschreibungskomplexität ist die Länge des Objekts selber. Es sollte also gelten:

$$K_f(s) \leq l(s)$$

Mit anderen Worten: Ein Beschreibungssystem, das als Grundlage für die Beschreibungskomplexität dient, sollte **komprimierend** sein.

Das Beschreibungssystem $\langle\{1, 2, 3, 4, 5\}, \{001, 010, 011, 100, 101\}, \{(001, 1), (010, 2), (011, 3), (100, 4), (101, 5)\}\rangle$ ist **nicht** komprimierend.

Beschreibungskomplexität: untere Grenze (lower bound)

Die natürliche untere Grenze der Beschreibungskomplexität ist trivialerweise die Länge 1:

$$K_f(s) \geq 1$$

Für jede Objektmenge, auch eine unendliche, lässt sich ein **trivial optimales** Beschreibungssystem angeben, falls

$$|S| = |\Sigma| \text{ mit } P \subset \Sigma^+$$

Natürlich wollen wir im Folgenden annehmen, dass

$$|S| > |\Sigma|$$

Beschreibungskomplexität und Formale Sprachen

Das Wort $\overline{w}$ einer Sprache L lizensiert durch eine Grammatik G oder Automaten M lässt sich beschreiben durch:

- Ableitungen/Parsebäume bei Grammatiken

$$K_G(\overline{w}) = |S \vdash \ldots \vdash \overline{w}|$$

- Zustandssequenz bei Automaten

$$K_M(\overline{w}) = TIME_M(\overline{w}) \geq |\overline{w}|$$

Beschreibungskomplexität und Formale Sprachen

Das Wort $\overline{w}$ einer Sprache L lizensiert durch eine Grammatik G oder Automaten M lässt sich beschreiben durch:

- Ableitungen/Parsebäume bei Grammatiken

$$K_G(\overline{w}) = |S \vdash \ldots \vdash \overline{w}|$$

- Zustandssequenz bei Automaten

$$K_M(\overline{w}) = TIME_M(\overline{w}) \geq |\overline{w}|$$

Das ist nur eine sehr grobe Annäherung. Adäquater wäre bestimmt

- bei K_G die verwendeten Regeltypen mit einzubeziehen (spricht die Entropie des Parsebaums)
- bei K_M den Speicherverbauch zu beachten (bzw. die Konfigurationen)

Outline

Kolmogorov-Komplexität: Idee

Ziel: Charakterisierung der inhärenten Beschreibungskomplexität von Objekten in potentiell unendlichen Objektmengen.

Idee: Charakterisierung mittels eines Programms, das das Objekt ausgibt. (Kolmogorov 1965)

Deshalb wird die Kolmogorov-Komplexität auch “algorithmische” Komplexität genannt.

Kolmogorov-Komplexität: Beispiel

Folge von 1000 Nullen (aus Wikipedia)

```
begin
  for i:= 1 to 1000
  print "0"
end
```

Die Folge aus 1000 Nullen ist also **komprimierbar**:

```
0begin00000000000000000000000000000000000000000
00for i:= 1 to 1000 0000000000000000000000000000
000print "0"0000000000000000000000000000000000000
0end00000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000
```

Kolmogorov-Komplexität: Universelle Turingmaschinen

Nur das kürzeste Programm zählt:

$$K_f(x) = \min_{f(p)=x} l(p)$$

Kolmogorov-Komplexität: Universelle Turingmaschinen

Nur das kürzeste Programm zählt:

$$K_f(x) = \min_{f(p)=x} l(p)$$

Problem: Es gibt unendlich viele, beliebig lange Programme, die ein Objekt ausgeben.

- Wie erreicht man Vergleichbarkeit?
- Welche Programmiersprache und Interpreter? (Java, Python, …)
- Welche Konstrukte innerhalb der Programmiersprache?

Kolmogorov-Komplexität: Universelle Turingmaschinen

Nur das kürzeste Programm zählt:

$$K_f(x) = \min_{f(p)=x} l(p)$$

Problem: Es gibt unendlich viele, beliebig lange Programme, die ein Objekt ausgeben.

- Wie erreicht man Vergleichbarkeit?
- Welche Programmiersprache und Interpreter? (Java, Python, …)
- Welche Konstrukte innerhalb der Programmiersprache?

Generalisierung: Universelle Turingmaschine T_U, die den Interpreter und das Programm (jeweils im **passenden** binären Format) als Eingabe hat.

$$T_\phi \# p \# \vdash^* T_\phi \# p \# x$$

Kolmogorov-Komplexität: Invariance Theorem

Bei einer T_U wird dann die Länge der Kodierungen des Interpreters $T\phi$ und eines Programms p addiert:

$$K_U(x) = \min_{T_\phi(p)=x} l(T_\phi) + l(p)$$

Wichtig ist, dass mit K_U die **absolute** Beschreibungskomplexität eines Wortes approximiert werden kann.

Kolmogorov-Komplexität: Invariance Theorem

Bei einer T_U wird dann die Länge der Kodierungen des Interpreters $T\phi$ und eines Programms p addiert:

$$K_U(x) = \min_{T_\phi(p)=x} l(T_\phi) + l(p)$$

Wichtig ist, dass mit K_U die **absolute** Beschreibungskomplexität eines Wortes approximiert werden kann.

Invariance theorem

$$\exists U \, \forall \phi \, \forall x : \quad K_U(x) \leq K_\phi(x) + c_\phi$$

- ϕ: berechenbare partielle Funktion mit $\phi(p) = x$
- c_ϕ: Länge der Kodierung von ϕ in U
- U: optimale universelle Turingmaschine

Kolmogorov-Komplexität: Invariance Theorem

Bei einer T_U wird dann die Länge der Kodierungen des Interpreters $T\phi$ und eines Programms p addiert:

$$K_U(x) = \min_{T_\phi(p)=x} l(T_\phi) + l(p)$$

Wichtig ist, dass mit K_U die **absolute** Beschreibungskomplexität eines Wortes approximiert werden kann.

Invariance theorem

$$\exists U \, \forall \phi \, \forall x : \quad K_U(x) \leq K_\phi(x) + c_\phi$$

- ϕ: berechenbare partielle Funktion mit $\phi(p) = x$
- c_ϕ: Länge der Kodierung von ϕ in U
- U: optimale universelle Turingmaschine

Trotzdem: $K_U(x)$ für beliebiges x ist nicht berechenbar, denn die Programme können Schleifen enthalten.

Beschreibungskomplexität und Kolmogorov-Komplexität: Abschließende Fragen

Beschreibungen von unendlichen Objekten wie formalen Sprachen?

- Nicht bei der Kolmogorov-Komplexität, weil die Universelle Turingmaschine nie halten würde.

Verhältnis von Kolmogorov-Komplexität zu Shannon'scher Entropie?

- sehr eng [2: §2.8.1]
- aber Entropie ist sehr viel leichter zu berechnen (siehe nächste Woche!)

Literaturverzeichnis

[1] Kolmogorov, Andrei N. 1965. Three approaches to the quantitative definition of information. *Problems of information transmission* 1(1). 1–7.

[2] Li, Ming & Paul Vitányi. 1997. *An introduction to Kolmogorov complexity and its applications*. 2nd (Graduate texts in computer science). New York: Springer.