Satz Gegeben eine kf Gramm. G, dann ist es unentscheidbar, ob

(1) G ist mehrdeutig,
(2) $\overline{L(G)}$ kontextfrei ist,
(3) $\overline{L(G)}$ regulär ist,
(4) $L(G)$ deterministisch kontextfrei ist.

$S \to S_1 \mid S_2$
$S_1 \to \ldots$ "G_1"
$S_2 \to \ldots$ "G_2"

Bew:

(1) Gegeben G_1 und G_2 wie im letzten Beweis. Sei G_3 die Gram. mit $L(G_3) = L(G_1) \cup L(G_2)$. (Das PCP K hat eine Lösung gdw. $L(G_1) \cap L(G_2) \neq \emptyset$. D.h. wenn K eine Lösung hat, dann ex. ein Wort w mit $w \in L(G_1)$ und $w \in L(G_2)$. D.h. G_3 ist in diesem Fall mehrdeutig.

(2) Seien G_1' und G_2' wieder (wie im letzten Beweis) die Gram. mit $L(G_1') = \overline{L(G_1)}$ und $L(G_2') = \overline{L(G_2)}$ (konstruierbar da G_1 und G_2 det. kf). Sei G_4 die Gram mit $L(G_4) = L(G_1') \cup L(G_2')$

Dann hat das PCP K eine Lösung folgw. $\boxed{L(G_1) \cap L(G_2)} =$

$\overline{\overline{L(G_1)} \cup \overline{L(G_2)}} = \overline{L(G_1') \cup L(G_2')} = \overline{L(G_4)}$ ist nicht kf.

(3+4) D.h. wenn das PCP K keine Lösung hat, dann ist $\boxed{L(G_4) = \Sigma^*}$, was eine reguläre (und det. kf.) Sprache ist.
Wenn das PCP K eine Lösung hat ist $L(G_4) = \Sigma^* - (L(G_1) \cap L(G_2))$.
$\overline{L(G_4)}$ ist dann nicht regulär und nicht det. kf. (da ja nicht kf). D.h. $L(G_4)$ kann nicht regulär sein (da reg. und det. kf. Spr. abgeschlossen unter Komplement).

□

Satz Gegeben eine kf. Sprache L_1 und eine reg. Spr. L_2, so ist unentscheidbar ob $L_1 = L_2$.

Bew: Folgt aus letztem Beweis zu (3+4). □

Satz Das Leerheitsproblem und das Endlichkeitsproblem für Typ 1 Sprachen unentscheidbar.

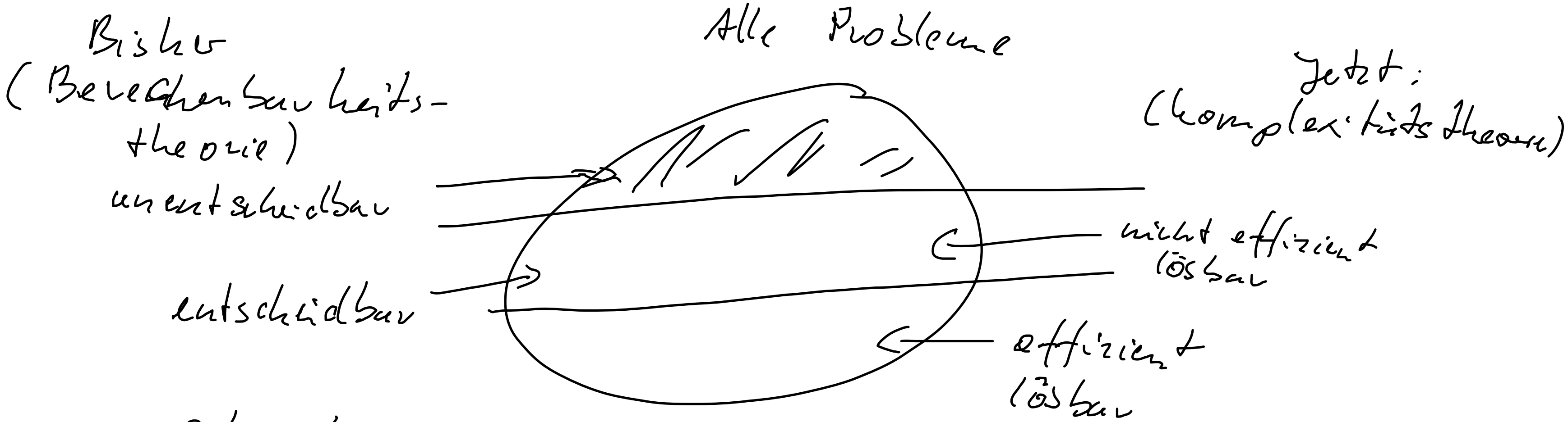
Bew: Wir reduzieren das Schnittproblem für kf. Spr. auf das Leerheitsproblem für Typ 1 Spr. Da Typ 1 - Spr effektiv unter Schnitt abgeschlossen sind ex. Typ 1 Gram. G_3 mit $L(G_3) = L(G_1) \cap L(G_2)$ (G_1 und G_2 kf).

$\square$

3.10 Gödelscher Unvollständigkeitssatz.

4. Komplexitätstheorie

Welche Berechnungsressourcen (Zeit, Speicherplatz) sind für das Lösen eines Problems notwendig?



Obere Schranken:
konkrete Algorithmen
(Bsp. CYK-Alg.)

Untere Schranken:
was brauchen wir mindestens?
(über alle möglichen Alg.)

Maschinenmodelle:
det. und nicht-det. Maschine
⇝ führt zu dem P=NP-Problem
⇝ eines der 7 Millenium-Probs. des Clay Mathematics Institute
⇝ 1 Millionen Dollar-Preise.

4.1 Komplexitätsklassen

Def $time_M : \Sigma^* \to \mathbb{N}$ ist eine Fkt, die die Anzahl der Rechenschritte einer det. TM M auf der Eingabe $x \in \Sigma^*$ angibt.

Def Sei $f: \mathbb{N} \to \mathbb{N}$ eine Fkt. Die Klasse $TIME(f(n))$ enthält alle Spr. A, für die es eine det. Mehrband-TM M mit $A = T(M)$ und $time_M(x) \leq f(|x|)$ gibt.

Bsp: $TIME(n) \supseteq$ Typ 3-Spr. $\quad TIME(c \cdot n^3) \supseteq$ Typ 2

Bem: Alle Mehrband-TMs mit Laufzeitbeschränkung $f(n)$ sind auf einer 1-Band TM mit $O((f(n))^2)$ simulierbar.

Def Ein Polynom ist eine Fkt. $p: \mathbb{N} \to \mathbb{N}$ der Form

$$p(n) = a_k \cdot n^k + a_{k-1} n^{k-1} + \ldots + a_1 \cdot n + a_0 \;, \quad a_i \in \mathbb{N}, \; k \in \mathbb{N}$$

Def Die Komplexitätsklasse P ist def wie folgt

$$P = \bigcup_{p \text{ Polynom}} TIME(p(n))$$

Bem: Nachweis, dass eine Spr. in P liegt: Alg. angeben, Laufzeit durch Polynom abschätzen.

Bsp: Wortproblem für kf Sprache.

Bem: Alg. mit $n \cdot \log n$ Laufzeit ist auch in P, da er durch n^2 abgeschätzt werden kann. Laufzeiten $\underline{n^{\log n}}$, $\underline{2^n}$ können nicht durch Polynome abgeschätzt werden.

Bem: Die Probleme, die in P liegen, werden als <u>effizient lösbare</u> Probleme angesehen. Theoretisch n^{1000} denkbar, in der Praxis meist $k \leq 5$.

Bem: Probleme in $TIME(2^n)$ oder $TIME(2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\}^n)$

sind immer noch primitiv rekursiv bzw. LOOP-berechenbar.

Begründung: Sei TM M mit $time_M(x) \leq f(|x|)$, wobei

f LOOP-berechenbar ist. Wandle TM in ein WHILE-Prog. mit einer WHILE-Schleife um:

WHILE $PC \neq 0$ DO ... END.

Ersetze diese durch

$y = f(n)$; LOOP $\underline{y}$ DO IF $PC \neq 0$ THEN ... END; END;

Bem: P ist relativ robust gegen Änderung des Maschinenmodells. Allerdings ist die Größe der dargestellten Zahlen manchmal wichtig:

```
INPUT(n);
x := 2;
LOOP n DO x := x * x END;
OUTPUT(x);
```

$x = \underline{2^{2^n}}$

0 Durchgänge	:	2
1 "	:	$4 = 2^2$
2 "	:	$16 = 2^{2^2}$
3 "	:	$256 = 2^{2^3}$
⋮		

Im weiteren wollen wir sehr große Zahlen ignorieren.