14 | SEMANTIK UND KOMPLEXITÄTSTHEORIE

> Ein überraschend kleines Fragment der Deduktionsfähigkeiten eines idealen Agenten scheint zu erfordern, dass selbst für eine finite Menge einfacher Fälle Ressourcen benötigt würden, die größer wären als die eines idealen Computers, der aus dem ganzen Universum konstruiert würde.
>
> (Cherniak 1986: 81)

1. Grundlagen
2. Einige wichtige Theoreme, die semantische Sachverhalte betreffen
3. Modellierung semantischer Verhältnisse durch Modallogik
4. Modellierung semantischer Verhältnisse durch *Description Logics*
5. Gehirnleistungsgrenzen?

§1 Grundlagen

Die Komplexitätstheorie beschäftigt sich nicht damit, was überhaupt berechen*bar* ist, sondern damit, welche Ressourcen Berechnungen erfordern. Zum einen geschieht dies durch sehr konkrete Abschätzungen des Aufwandes von Algorithmen, zum anderen werden aber sehr grobe Komplexitätsklassen (von Problemen bzw. zu generierenden Sprachen) unterschieden, da selbst in dieser – für unser naives Verständnis extremen – Grobheit der Abschätzung wichtige Erkenntnisse über die unterschiedliche Schwierigkeit der Bewältigung zweier Probleme und ihres diesbezüglichen Abstandes gewonnen werden können.

Standardmäßig betrachtet werden Zeit- und Raumkomplexität. Raumkomplexität betrifft den Speicherbedarf, der irgendwann während der Rechnung von einem Algorithmus benötigt wird. Zeitkomplexität betrifft die Anzahl der Rechenschritte, die der Algorithmus läuft, wenn er zu einem Ergebnis führt. (Algorithmen, die partielle Funktionen berechnen, sind auf

einigen Eingabewerten nicht definiert und erreichen dann nie einen Endzustand [laufen unendlich]).

Das Komplexitätsmaß wird berechnet relativ zur Länge der Eingabe. Eingaben werden typischerweise binäre codiert (d.h. 8, 9, 10 .. 15 haben die Länge 4: 1000, 1001, 1010 .. 1111). Maschinenmodelle abstrakter Automaten können sich dadurch unterscheiden, wie viele Zeichen das Inputalphabet Σ umfasst und wie viele Arbeitsbänder etwa eine Turing-Maschine besitzt. Es lässt sich jedoch zeigen, dass eine Kanonisierung auf ein Inputalphabet mit 2 Zeichen „0“ und „1“ (und einem Hilfszeichen „#“) sowie auf 2 Arbeitsbänder mit höchstens quadratischer Verzögerung möglich ist. Das heißt: Wird eine deterministische Maschine M eines Typs (sei es eine Registermaschine, sei es eine Turing-Maschine mit mehr Arbeitsbändern ...) durch eine deterministische Turing-Maschine (DTM) simuliert bzw. dasselbe Problem von einer DTM M’ berechnet, so geschieht dies in einer Zeit, die höchstens quadratisch größer ist als der Zeitbedarf der ursprünglichen Maschine. In Zahlen ausgedrückt kann dies einen großen Unterschied machen (etwa 300 gegenüber 90000), bezüglich der groben und großen Abschätzungen, welche bei semantischen Problemen auftreten, kann dies indessen vernachlässigt werden! Die konkreten Rechnungsleistungen von Speicherchips und CPUs sind deshalb auch zunächst nicht relevant. Σ sei also das Inputalphabet, Wörter sind Abfolgen „011010..“, ε sei das leere Wort.

Ausgangspunkt sind Turing-Maschinen (TMs). Betrachtet werden nicht nur DTMs, sondern auch Nichtdeterministische Turing-Maschinen (NTMs). Bei einer NTM lässt deren Überführungsfunktion zu, dass gegeben einen Input und einen Zustand mehr als ein Nachfolgezustand angenommen werden kann. Auch hier kann eine Kanonisierung auf 2 mögliche Nachfolgezustände erfolgen. Ob man NTMs bauen kann, ist zunächst nicht von Interesse. Diese Art von Maschinen wird eingeführt, um Komplexitätsklassen unterscheiden zu können. (NTMs können nicht *mehr* berechnen als DTMs, sie unterscheiden sich aber bezüglich der Berechnungskomplexität.)

Eine DTM akzeptiert einen Input, wenn die Berechnung in einem Akzeptanzzustand hält. Eine NTM akzeptiert einen Input, wenn es mindestens eine Berechnung (aus dem durch den Nichtdeterminismus gegebenen Baum möglicher Berechnungen der NTM einen Ast) gibt, die in einem Akzeptanzzustand hält.

Es gibt zwei Arbeitstypen einer Maschinenart: *Akzeptoren* sind Maschinen, die einen Input prüfen, ob er zu der von ihnen erzeugten Wortmenge (der

von ihnen erzeugten Sprache) gehört. Hält die Maschine in einem Akzeptanzzustand gehört der Input zur erzeugten Sprache, sonst nicht. *Transducer* sind Maschinen, die gegeben einen Input einen Output auf ein Ausgabeband schreiben. Die Wörter, die bei der Annahme eines Akzeptanzzustandes auf dem Outputband stehen, gehören zur erzeugten Sprache. Zu einer Maschine M gehört die von M erzeugte/akzeptierte Sprache A = L(M). Eine Maschine berechnet so eine Sprache A bzw. löst Vorkommnisse des *Problems* "x∈ A?".

Das Komplexitätsmaß bezüglich einer Inputlänge ist bei DTMs nur dann definiert, wenn die DTM bei allen Inputs dieser Länge hält. Bei NTMs reicht es, dann es für jeden Input eine Berechnung gibt, die hält; d.h. die Komplexität für NTMs bezüglich einer Inputlänge ist definiert, wenn es für jeden Input dieser Länge mindestens eine Berechnung gibt, die auf dem Input hält.

Bezüglich einer DTM ist das Komplexitätsmaß auf Inputs der Länge x die größte Zahl an Schritten und benötigtem Speicher, die bei den Berechnungen auf dieser Länge auftreten kann (d.h. es wird immer der *worst case* betrachtet).

Bezüglich einer NTM ist das Komplexitätsmaß auf Inputs der Länge x die größte Zahl an Schritten und benötigtem Speicher, die bei Berechnungen auf dieser Länge auftreten kann, wobei bei jedem einzelnen Input einer Länge x die Berechnung in die Bestimmung des Komplexitätsmaßes eingeht, die *am wenigsten* Ressourcen benötigt (d.h. es zählt der schwierigste Input einer Länge x, aber bezüglich seiner die schnellstmögliche Berechnung).

Dass diese Definition „unfair“ gegenüber den DTMs sind mag sein, es geht allerdings ja um die Abgrenzung zwischen Komplexitätsklassen nicht um Ingenieurswettbewerbe.

Komplexitätsklassen werden durch Funktionen bzw. durch Klassen von Funktionen definiert:

- DTIME(f) ist die Menge der Probleme, die Zeit im Umfang von f(x) bei einem Input der Länge x bei einer DTM benötigen.
- NTIME(f) entsprechend für NTM.
- DSPACE(f) ist die Menge der Probleme, die Raum im Umfang von f(x) bei einem Input der Länge x bei einer DTM benötigen.
- NSPACE(f) entsprechend für NTM.

Einige spezielle Komplexitätsklassen sind von besonderem Interesse:

- LOG ist die Menge der Problem, die logarithmischen Raum bei DTMs benötigen.
- NLOG entsprechend für NTMs.
- P ist die Menge der Probleme, die x^c Zeit (polynome Zeit) bei einem Input der Länge x bei einer DTM benötigen.
- NP entsprechend für NTMs.
- EXT ist die Menge der Probleme, die 2^x (exponentielle Zeit) bei einem Input der Länge x bei einer DTM benötigen.
- NEXT entsprechend für NTMs.
- PSPACE ist die Menge der Probleme, die bei einer DTM polynomen Raum benötigen.
- NSPACE entsprechend für NTM.
- EXP ist die Menge der Probleme, die $2^{p(x)}$ Zeit (exponentielle Zeit mit polynomem Exponenten) bei Input der Länge x bei einer DTM benötigen.
- NEXP entsprechend für NTM.

Probleme, welche in die Klasse P fallen, gelten als praktisch lösbar. Probleme, die in DEXT fallen, übersteigen schnell (d.h. schon bei Eingaben recht beschränkter Länge) die praktisch zur Verfügung stehenden Ressourcen. Sie gelten daher als praktisch nicht berechenbar, selbst wenn sie bewiesenermaßen berechen*bar* sind. Ein Problem fällt in eine Problemklasse C, wenn ein Algorithmus bekannt ist, der dieses Problem löst und dessen Komplexität bezüglich einer ihn implementierenden Maschine in C fällt. Es handelt sich bei diesem Vorgehen also nur um Positivnachweise: Der Umstand, dass wir (zunächst) einen Algorithmus in DEXT finden, schließt nicht *per definitionem* aus, dass wir auch einen anderen in P finden, so dass das betrachtete Problem eigentlich praktisch lösbar ist.

Gegeben eine Komplexitätsklasse C ist co-C = $\{ \mathsf{E} \mid \bar{\mathsf{E}} \in C\}$ (d.h. die Menge der Probleme, deren Komplement in C ist).

Der besondere Witz von NTMs ist, dass man mittels des Nichtdeterminismus das *Raten* von Inputs einführen kann

```
y := ε
for x:= 1 to i do
    choose one
        y := y0;
        y := y1;
        y := y;
```

Algorithmus um ein Wort mit einer Länge ≤ i zu raten

Da Raten immer eine konstante Zeit einnimmt und konstante Faktoren in den Größenabschätzungen, welche die Komplexitätstheorie interessieren, vernachlässigt werden können, kann Raten als *ein* Schritt angenommen werden.

Wichtig ist dass es Probleme gibt, bei denen das Erzeugen eines Wortes einen sehr großen Aufwand mit sich bringt, jedoch der Test, ob ein Wort zur gesuchten Sprache gehört, wesentlich einfacher ist. Hier liegt der Vorteil von NTMs: Sie raten ein betreffendes Wort – das Raten kostet keinen Aufwand – und prüfen dann – mit einem gewissen Aufwand – ob das Wort zu der betreffenden Sprache gehört. DTMs müssten – im Vergleich dazu – den ganzen Aufwand betreiben, um das Wort zu erzeugen.

Deshalb wird im Allgemeinen davon ausgegangen, dass $P \neq NP$. Eine Reihe von Problemen, für die wir nur einen Algorithmus in DEXT für DTMs kennen, haben eine Lösung für NTMs in NP. Da sich NTMs nicht unmittelbar bauen lassen, hat dies keinen praktischen Nutzen (Simulationen/Implementationen von NTMs auf DTM-Basis benötigen exponentielle Zeit). Die Klasse NP dient so jedoch zur Abgrenzung der Probleme, die *gemäß unserem (besten) Wissen* praktisch nicht (leicht) lösbar sind.

Zum Vergleich von Problemen ist die Relation der Reduzierbarkeit zwischen Problemen wichtig. Ein Problem $x \in A$ ist auf ein zweites $y \in B$ reduzierbar genau dann, wenn es eine reduzierende Funktion f gibt, so dass gilt $x \in A \Leftrightarrow f(x) \in B$. Falls die reduzierende Funktion maximal polynome Zeit benötigt, handelt es sich um die Reduktionsrelation $\leq_m$ der polynomen Reduzierbarkeit. Ein Problem B ist *hart* für eine Komplexitätsklasse C genau, wenn für alle $A \in C$ gilt: $A \leq_m B$. B drückt dann paradigmatisch den Schwierigkeitsgrad dieser Klasse C aus. Ein Problem B ist *vollständig* für eine Komplexitätsklasse C genau, dann B C-hart ist und $B \in C$. Vollständige bzw. harte Probleme sind von besonderem Interesse, da sich

im Vergleich zu ihnen Aussagen über ein gerade untersuchtes Problem machen lassen. Gilt für das untersuchte Problem A in Vergleich zu einem NP-harten Problem B dass $A \leq_m B$ wissen wir, dass A (genauer immer: das Lösen einer Instanz "$x \in A$?") auf einer NTM nicht mehr als polynome Zeit benötigen würde; gilt $B \leq_m A$ heißt dies, dass A mindestens in NP fällt, also praktisch mutmaßlich kaum zu lösen ist. Bei solchen Nachweisen muss allerdings auch bedacht werden, ob die zu komplexen Eingabelängen (d.h. diejenigen, die auf den bekannten, evtl. exponentiellen Algorithmen die Ressourcen übersteigen) in den intendierten Anwendungen auftreten können oder werden.

Nach *Savitchs Theorem* lässt sich nicht-deterministischer Raum durch deterministischen Raum *per quadratischer Komplexitätssteigerung* simulieren [also NSPACE(f(|n|)) durch DSPACE(f(|n|)²)]. Daraus folgt unmittelbar PSPACE = NSPACE. Wir können zur Realisierung von PSPACE-vollständiger Probleme auch nicht-deterministische Algorithmen verwenden!

§2 Einige wichtige Theoreme, die semantische Sachverhalte betreffen

Fragen, die in der Logik typischerweise zur Semantik gezählt werden, sind solche der Prüfung auf Erfüllbarkeit, Gültigkeit, Ungültigkeit von Formeln eines Kalküls. Einige der hier einschlägigen Probleme sind die folgenden:

- CNF ist die Menge aller aussagenlogischen Formeln in Konjunktiver Normalform.
- SAT ist die Menge aller aussagenlogisch erfüllbaren Formeln. (Eine Prüfung "$x \in$ SAT?" fragt also, ob x eine erfüllbare Formel ist/codiert.)
- SAT-CNF ist die Menger aller aussagenlogischen Formeln in Konjunktiver Normalform, die erfüllbar sind.
- INCON ist die Menge aller aussagenlogischen Formeln, die unerfüllbar sind; INCON ist das Komplement zu SAT.
- VALID ist die Menge der gültigen aussagenlogischen Formeln.
- QBF ist die Menge aller quantifizierten aussagenlogischen Formeln *ohne* freie Aussagenbuchstaben, die als "wahr" bewertet werden. Quantifizierte aussagenlogische Formeln sind solche, bei denen es neben den Junktoren einen All- und einen Existenzquantor gibt, die sich auf Aussagenvariablen beziehen.

- QBF-VALID ist die Menge der gültigen quantifizierten aussagenlogischen Formeln (ohne ungebundene Aussagenbuchstaben).
- REAL+ ist die Menge der gültigen Formeln der Realzahlarithmetik mit Addition (ohne Multiplikation). Diese Menge ist entscheidbar.
- PRESB ist die Menge der gültigen Formeln der Presburger Arithmetik (Arithmetik der ganzen Zahlen ohne Multiplikation). Diese Menge ist entscheidbar.

Es gelten nun die folgenden Theoreme:

(T1) $CNF \in P$

Beweis[1]: Eine aussagenlogische Formel der Länge n hat weniger als n Junktoren. Die Umformung in KNF geht rekursiv vor (von Innen nach Außen bezüglich der durch Junktoren gebundenen Aussagenbuchstaben), d.h. die Rekursionstiefe ist maximal n (für höchstens jeden Junktor wird die Rekursion benötigt). Innerhalb der Rekursion können nur Formeln weiter verarbeitet werden, die kürzer als die Ausgangsformel sind, d.h. deren Länge ist $\leq n$. Das Laufen durch die Rekursion benötigt also maximal n^2 Zeit, die übrigen Schritte fallen nicht mehr ins Gewicht. Entsprechend lange dauert die Überprüfung, ob eine Formel in KNF ist. ■

(T2) $SAT \in DEXT$

Beweis: Eine DTM M erzeugt bezüglich einer Formel α alle möglichen Belegungen der sie aufbauenden Elementaraussagen (d.h. die Formel wird umgebaut, indem jeweils "1" oder "0" eingesetzt wird). Die Überprüfung, ob die so entstandene Formel erfüllbar ist, verfährt nach den (rekursiven) Wahrheitsbedingungen der Aussagenlogik. Analog zur Argumentation zu (T1) benötigt diese Überprüfung nur polynome Zeit. Die Generierung aller möglichen Belegungen bei n Elementaraussagen benötigt indessen exponentielle Zeit (2^n). (Man denke an die Zeilen einer entsprechenden Wahrheitstafel.) Eine Formel der Länge n besteht aus höchstens n Elementaraussagen. Also gibt es höchstens 2^n viele Belegungen, die erzeugt werden müssen. Die rechtlichen Schritte fallen dagegen nicht mehr ins Gewicht. ■

Das Problem einer DTM bezüglich SAT ist, dass im schlechtesten Fall alle Zeilen der Wahrheitstafel durchgegangen werden müssen, um die erfüllende Belegung zu finden. Dies entspricht der *Suche in die Breite*, die alle möglichen Lösungen durchgeht. Eine NTM M' hingegen kann eine Belegung raten – dies benötigt *per definitionem* nur einen Schritt – und in polynomer Zeit prüfen, ob die geratene Belegung einer erfüllende Belegung ist.

---

[1] Die formalen Beweise finden sich in der unten angegebenen Literatur.

Die schnellste akzeptierende Berechnung bezüglich α bei M', wenn α erfüllbar ist, benötigt daher nur polynome Zeit. Aufgrund der Definition der NTM-Zeitkomplexität ergibt sich:

(T3) SAT∈ NP

Da SAT∈ NP ist auch INVALID∈ NP (die Prüfung, ob eine Formel ungültig ist, d.h. falsch gemacht werden kann).

Um bezüglich einer quantifizierten aussagenlogischen Formel die Wahrheit festzustellen, sind neben den rekursiven Klauseln für die Junktoren die (üblichen) Klauseln für die Quantoren zu benutzen. Ein Quantor bezieht sich dabei [etwa in (∃X)(...∧ X ... ⊃ ...)] auf die Menge der möglichen Belegungen. Der *Zeit*aufwand einer entsprechenden Rekursion ist daher entsprechend zu dem oben bei SAT Gesagten bei einer DTM exponentiell. Allerdings kann, da bezüglich jeden Rekursionsschrittes nur das Ergebnis ("wahr" oder "falsch") interessiert, der für die Auswertung nötige *Platz* wiederverwendet/überschrieben werden. Zwischengespeichert werden müssen nur die Werte für die rekursiven Aufrufe. Der Platzbedarf kann daher den des Rekursionsumfangs (also n^2 [s.o.]) nicht überschreiten. Daher gilt:

(T4) QBF∈ PSPACE

Bei entscheidbaren Problemen von Theorien Erster Stufe ergibt sich die Zeitkomplexität durch den Ausgriff auf die Variablenbelegung bzw. die Interpretationen der Konstanten. Diese Probleme sind wesentlich härter als die aussagenlogischen Probleme. Beispielsweise gelten die folgenden Theoreme:[2]

(T5) REAL+∈ NEXT
(T6) PRESB∈ NEXP

Von besonderem Interesse sind im Weiteren SAT und QBF, da folgende Theoreme gelten:

(T7) SAT ist NP-hart (und vollständig).

(T8) QBF ist PSPACE-hart (und vollständig).

Beweisidee: Turing-Maschinen lassen sich auf verschiedene Weisen codieren. Eine Weise ist, jedem Zustand und jeder Bandzelle eine aussagenlogische Konstante zuzuordnen, sowie Konstanten dafür, dass der Kopf der Maschine auf der entsprechende Bandzelle steht. Die Konstante wird mit "1" bewertet, wenn etwa die Zelle mit "1" beschriftet ist oder sich die

---

[2] Zu den Beweisen vgl. (Hopcroft/Ullman 1979: 354-62).

Maschine im entsprechenden Zustand befindet. Der Konfiguration, in der sich eine TM befindet, entspricht dann eine komplexe aussagenlogische Formel (bzw. ein entsprechendes Binärwort). Die Bedingungen der Überführungsrelation der TM lassen sich durch Konditionale ausdrücken. Der Gesamtberechnung einer TM entspricht dann eine – sehr lange – Aussage, die besagt, dass von einer bestimmten Anfangskonfiguration aus eine (akzeptierende) Endkonfiguration erreichbar wird. Genau dann, wenn diese Aussage *erfüllbar* ist, akzeptiert die beschriebene TM den Input. ■

Lässt sich also von einem Problem A zeigen, dass $SAT \leq_m A$, dann ist A auch NP-hart, d.h. es bestehen Zweifel, ob A praktisch lösbar ist. Lässt sich von einem Problem A zeigen, dass $QBF \leq_m A$, dann ist A auch PSPACE hart, *kann* also nicht unterhalb exponentieller Zeit operieren (wenn $P \neq NP$).

Da für den Fall, dass E C-hart ist, gilt dass Ē co-C hart ist, und man davon ausgehen muss, dass INCON exponentielle Zeit benötigt, da alle Belegungen durchgegangen werden müssen, um die Unerfüllbarkeit einer Formel zu erweisen, ist davon auszugehen, dass NP ≠ co-NP (d.h. in co-NP sind exponentielle Probleme zu erwarten, während P = co-P). Wenn eine Formel inkonsistent ist, ist ihre Negation gültig. Eine TM M', die bis auf die Umkehrung von Akzeptanz/Zurückweisung (im letzten Schritt) mit einer Maschine M, die INCON berechnet, übereinstimmt, berechnet also VALID. Nach dem Gesagten ist anzunehmen dass VALID∉ NP, d.h. für die Aussagenlogik fallen die Komplexität von Erfüllbarkeit und Gültigkeit in verschiedene Komplexitätsklassen.

Da QBF schon, bei der rekursiven Auswertung der Wahrheitsbedingungen der Quantoren auf die Gesamtheit der Variablenbelegungen ausgreift, kommt mit QBF-VALID keine weitere Komplexitätssteigerung mehr hinzu. Es gilt auch hier:

(T9) QBF-VALID ist PSPACE-vollständig.

Für die quantifizierte Aussagenlogik fallen also die Prüfung der Erfüllbarkeit/Wahrheit einer Aussagen und die Prüfung ihrer Gültigkeit in dieselbe Komplexitätsklasse.[3]

[3] Da sich QBF allein auf Formeln ohne freie Aussagenbuchstaben bezieht, fallen hier Wahrheit und Erfüllbarkeit zusammen.

§3 Modellierung semantischer Verhältnisse durch Modallogik

Semantische Beziehungen werden oft modallogisch dargestellt, da sich der Unterschied zwischen wahrheitsfunktionalen Konditionalen und semantischem Enthaltensein durch Modaloperatoren ausdrücken lässt. Betrachtet werden im Folgenden *normale* Modallogiken (von **K** bis **S5**). Man kann mit diesen semantische Beziehungen ausdrücken durch den Unterschied zwischen den beiden Sätzen:

(1) $(\forall x)(\text{Wohnungskatze}(x) \supset \text{BekommtDosenfuttter}(x))$

(2) $\Box(\forall x)(\text{Wohnungskatze}(x) \supset \text{Säugetier}(x))$

Modallogische Sprachen sind dann in der Regel Erweiterungen der Aussagen- oder der Prädikatenlogik. (Wir betrachten hier nur Aussagenlogiken.) Damit ist unmittelbar klar:

(T10) Das Problem der Erfüllbarkeit normaler Modallogiken ist NP-hart.

Beweis: Da normale Modallogiken die Aussagenlogik erweitern, enthalten sie alle aussagenlogischen Formeln; und deren Erfüllbarkeitsproblem SAT ist (schon) NP-hart. ■

Es stellen sich damit zwei Fragen:

1. Ist das Erfüllbarkeitsproblem normaler Modallogiken noch komplexer/härter als das der Aussagenlogik?
2. Ist das Erfüllbarkeitsproblem verschieden komplex/hart für verschieden starke Modallogiken (beispielsweise bezüglich des Vergleichs des stärkeren Systems **S5** gegenüber **K**)?

(ad Frage 1)

Eine Reihe von normalen Modallogiken haben die Eigenschaft Finiter Modelle (*finite model property*). Die Logiken, die wir hier betrachten sind zudem finit axiomatisierbar und machen nicht von Regeln Gebrauch, die nicht-finite Prämissen haben. Dadurch werden diese Logiken Λ entscheidbar: Man nehme zwei DTMs M und M*: M zählt die Λ-Theoreme auf, M* zählt die finiten Λ-Modelle auf. Bezüglich der Gültigkeit einer beliebigen Λ-Formel α wird entweder M α generieren (so dass α gültig ist) oder M* generiert irgendwann eines der α falsifizierenden Modelle. Λ hat die *Polynome Modeleigenschaft* wenn bezüglich einer Modellmenge *M* jede Λ-Formel φ erfüllbar ist in einem Modell aus *M,* das nur $p(|\varphi|)$ viele Zustände/Welten hat, für ein Polynom *p*. Die Polynome Modelleigenschaft

alleine garantiert nicht die Entscheidbarkeit, da wir das betreffende Modell auch finden (können) müssen. *M* muss als rekursiv angenommen werden. Dann ergibt sich die Entscheidbarkeit. Die finiten Modell lassen sich anlässlich der Überprüfung einer mutmaßlichen Konsequenzbeziehung generieren per Filtration bezüglich der relevanten Elementaraussagen. Eine NTM kann eine solche Filtration *raten*! Die Frage ist, wie viele Welten sind in einem solchen finiten Modell befinden (müssen), relativ zur Länge n der Inputformel.

Angenommen eine Logik Λ hat die Polynome Modelleigenschaft. Dann kann eine NTM M in einem Schritt ein solches Modell raten und in polynom vielen Schritten [relativ zur Länge $|\varphi|$ der Inputformel] überprüfen, ob dieser Modell φ verifiziert, *wenn* wir die üblichen rekursiven Wahrheitsbedingungen für die extensionalen Junktoren zugrunde legen [s.o.] *und* die Wahrheitsbedingungen für die Modaloperatoren uns nur auf die Hinzunahme polynom [$p(|\varphi|)$] vieler Welten zwingen. (Neue Welten müssen wir – bekanntlich – nur durch die $\Diamond$-Bedingung hinzufügen.)

Eine solche Logik Λ hätte damit ein Erfüllbarkeitsproblem SAT-$\Lambda \in$ NP, wäre also NP-vollständig. Und damit wäre diese Modallogik Λ *nicht* härter/komplexer als die Aussagenlogik.

(ad Frage 2)

Welche normale Modallogik ist nun eine solche Logik Λ vom gerade beschriebenen Typ? **S5** ist eine solche Logik! Denn in **S5** können alle modalen Sätze auf den Modalgrad 1 reduziert werden. Es gibt dann keine iterierten Modalitäten mehr. Neue Welten werden nur die $\Diamond$-Operatoren gefordert. Jeder $\Diamond$-Operator fordert indessen nur *eine* weitere Welt, d.h. gegeben, dass in einer Formel der Länge n immer weniger als n $\Diamond$-Operatoren vorhanden sind ist die Zahl der benötigten Welten *linear* in der Länge der Inputformel. **S5** besitzt also die Polynome Modelleigenschaft. Also:

(T11) SAT-S5 ist NP-vollständig.

Die deduktiv stärkste der gängigen normalen Modallogiken bringt also keine Komplexitätssteigerung mit sich! Wenn φ nicht **S5**-erfüllbar ist, ist $\neg\varphi$ gültig. VALID-S5 ist somit so komplex wie das Komplement von SAT-S5, also co-NP-vollständig. Damit fallen für normale Modallogiken Λ des Typs von **S5** Erfüllbarkeit und Gültigkeit in verschiedene Komplexitätsklassen [s.o.]. Nach *Bulls Theorem* haben alle normalen Modallogiken ab **S4.3** die Finite Rahmeneigenschaft, so dass für sich für sie dasselbe

ergibt wie für **S5**.

Normale Modallogiken *unterhalb* von **S4** haben hingegen eine Fülle nicht reduzierbarer Modalitäten. Insbesondere die einfache normale Modallogik **K** hat *nicht* die Polynome Modelleigenschaft, obwohl die Modelle finit sind! Tatsächlich besagt *Ladners Theorem*

(T12) Für jede normale Modallogik Λ von **K** bis **S4** (einschließlich) ist SAT-Λ PSPACE-hart.

Da davon auszugehen ist, dass PSPACE $\neq$ NP, aber gilt NP $\subseteq$ PSPACE, sind die schwächeren Modallogiken also komplexer als die stärkeren. Ihr Erfüllbarkeitsproblem ist härter als das der Aussagenlogik und praktisch zeitlich kaum zu realisieren (auf längeren Eingaben).

Beweisidee:[4] Man kann eine sehr lange Konkjunktion φ konstruieren, so dass die nicht reduzierbaren Modalitäten in der Entwicklung eines Modells für φ einen *vollständigen Binärbaum* erzwingen, wodurch das Modell zu φ zwar finit aber exponentiell ist. Bezüglich jeder Zahl n lässt sich eine solche Formel φ konstruieren. Ihre wesentlichen Teilformeln sind von den folgenden beiden Formen (für $0 \leq i \leq n$):[5]

(3) $\Box^i(q_i \supset (\Diamond(q_{i+1} \wedge p_{i+1}) \wedge \Diamond(q_{i+1} \wedge \neg p_{i+1})$

(4) $\Box^i((p_i \supset \Box p_i) \wedge (\neg p_i \supset \Box \neg p_i)$

wobei es n viele Konjunkte jeder dieser beiden Formen gibt. Die Konjunkte der Form (3) erzwingen einer Verzweigung. Und die Konjunkte der Form (4) schreiben fest, dass diese Verzweigungen (im Baum) vererbt werden. Ein Modell für φ mit $m = |\varphi|$ muss dann 2^m viele Welten enthalten. Die Überprüfung, ob ein geratenes Modell φ verifiziert, bedarf daher exponentieller Zeit (d.h. höchstens $\mathbf{K} \in$ NEXT). Hingegen kann man schrittweise (pro Ast des Binärbaumes) prüfen, ob das geratene Modell die entsprechenden Teilformeln erfüllt. Pro Ast benötigt diese Prüfung nur polynomen Raum, da $|\varphi|$ polynom in der Zahl n der höchsten Modalitäteniteration ist und so der Baum nur polynome Tiefe hat. Entsprechende einer Suche *depth first* kann eine NTM M Ast für Ast ein geratenes Modell auswerten: Sie merkt sich das Ergebnis für den Ast (dies benötigt für alle Äste nur polynomen Platz) und überschreibt mit der Berechnung für den nächsten Ast den vorhandenen Raum (ebenfalls polynom). Also ist SAT-

---

4 Zur formalen Durchführung siehe (Blackburn/de Rijke/Venema 2001: 381-92).

5 Erinnert sei daran, dass z.B. in **K** die Modalitäten nicht per Distribution und Äquivalenzen wie $\Box\Box A \equiv \Box A$ reduziert werden können.

K∈ PSPACE. Für **S4** kann man (durch eine Art Umkehr des gerade betrachteten Verfahrens der Formelkonstruktion ausgehend von Bäumen, die sich durch die Analyse von QBF-Formeln ergeben) zeigen, dass QBF-VALID sich auf **S4**-Erfüllbarkeit reduzieren lässt. ■

Da die Überprüfung der Erfüllbarkeit schon den ganzen Binärbaum durchlaufen muss, ergibt sich – wie bei QBF – kein Unterschied zwischen Erfüllbarkeits- und Gültigkeitsproblemen bei diesen Logiken, sie fallen beiden in die Komplexitätsklasse PSPACE.

Eine Reihe temporaler Modallogiken haben eine entsprechende Härte. Als Hintergrund für den nächsten Abschnitt seien noch Dynamische Logiken erwähnt.

Dynamische Logiken haben bezüglich einer Menge Π von Handlungen eine Menge von Modalitäten $[\pi_i]$ für Notwendigkeit. Diese Modalitäten sind – natürlich – nicht reduzierbar und von einander verschieden. Also ist die Erfüllbarkeitsfrage der Standard Dynamischen Logik **PDL** mindestens so hart wie die bei **K**! Hinzukommt allerdings, dass es in **PDL** auch den Operator "*" für den transitiven Abschluss einer Reihe von Aktivitäten gibt, etwa $[\pi_i]^*$. $[\pi_3]^*A$ besagt, dass nach *beliebiger Wiederholung* von π_3 immer A der Fall ist. Damit ergibt sich eine wesentlich höhere Komplexität. **K**-Erfüllbarkeit war deshalb "nur" PSPACE-hart, weil der betreffende Algorithmus astweise vorgeht und dabei den Speicher wiederverwendet. Bei **PDL** gilt hingegen, dass ein entsprechender Berechnungsbaum Äste exponentieller Länge aufweisen kann. Es ergibt sich

(T14) **PDL**-SAT∈ EXT.

Beweisidee:[6] Bei Spielen kann man nach Gewinnstrategien suchen. Diese Suche antizipiert alle möglichen Antworten des Gegners auf einen eigenen Zug. Die Kriterien, eine Gewinnstrategie zu finden, arbeiten rückwärts vom gewonnen Spiel her und legen fest, dass nach beliebig vielen Zügen die gewinngarantierenden Eigenschaften eines ersten Zuges noch vorhanden sind. Diese Vererbung über *beliebig viele* Züge entspricht dem "*"-Operator. Mittels seiner lassen sich die Kriterien für Gewinnstrategien in **PDL** formalisieren. Also sind Gewinnstrategien Modelle für Formeln von **PDL**. Also ist **PDL**-Erfüllbarkeit mindestens so hart wie das Finden von Gewinnstrategien, welches exponentiell ist. ■

---

6 Zur formalen Durchführung siehe (Blackburn/de Rijke/Venema: 393-405).

§4 Modellierung semantischer Verhältnisse durch Description Logics

Das Problem der üblichen Semantikmodellierung durch Modallogiken scheint zu sein, dass es sich bei diesen Modallogiken in der Regel um Erweiterungen der Standardlogiken handelt. Die Komplexität der betrachteten Systeme kann also nicht abnehmen. Da NP-Härte praktische Undurchführbarkeit – zumindest auf hinreichend langen Eingaben – mit sich bringt, scheint dies ein Mangel der Modellierung zu sein. Es kann sich daher lohnen, spezielle Formalismen zu betrachten, die quer zu der Frage Erweiterung/Äquivalenz im Vergleich zu den Standardlogiken stehen.

Eine Gruppe solcher Formalismen sind die *Description Logics* (DSL). Es handelt sich hier um Formalismen, die – anders als Modallogiken – direkt aus den Kognitionswissenschaften (hier insbesondere Theorien der Wissensrepräsentation und der Künstlichen Intelligenz) hervorgegangen sind.

DSLs repräsentieren semantische Relationen durch einen Formalismus, der aussagenlogische Verknüpfungen von Kategorien/Begriffen verbindet mit Quantorenausdrücken, die sich allerdings nicht auf Formeln mit Variablen oder Konstanten beziehen. Unterschieden wird dazu zwischen atomaren und zusammengesetzten Begriffen und *Rollen*. Das basale Verhältnis zwischen Begriffen ist das der *Subsumption*[7]: $A \sqsubseteq B$. Per wechselseitiger Inklusion ergibt sich die *Gleichheit*, die für Definitionen $A \equiv B$ benötigt wird. *Erfüllbarkeit* heißt bei einem Begriff, dass er nicht leer ist. Zusammengesetzte Begriffe drücken den Schnitt zweier Begriffe aus "$A \sqcap B$" oder deren Vereinigung "$A \sqcup B$". Es kann mit "$\neg B$" das Komplement eines Begriffes gebildet werden. Rollen betreffen das Verhältnis von Begriffen: "$\forall R.C$" verlangt, dass alle Individuen, die in der Relation R zu Individuen stehen, die unter einen – in der Formel nicht genannten – Ausgangsbegriffs stehen, C sind. Entsprechend "$\exists R.C$". Quantoren können auch numerisch beschränkt eingesetzt werden, wie "≥ 3 hasChild" ausdrückt mindestens drei Kinder zu haben. Existenzquantifikation ergibt sich als $\exists R.\top$. Rollenkonstruktoren erlauben Schnitt, Komplement, Vereinigung, aber auch transitive Hülle und Inverse einer Rolle (d.h. einer Relation) zu definieren.

Der Begriff "Elternteil" wird also ausgedrückt:

$$\exists \text{hasChild.Person} \sqcap \forall \text{hasChild.Person}$$

---

7 Gemeint ist offensichtlich Subordination.

Jemand ist ein Elternteil, wenn er/sie ein Kind hat, das eine Person ist, und nur Personen als Kinder hat. Auch Rollen können aus atomaren Begriffen und Rollen aufgebaut werden, z.B. "höchstens zwei Töchter haben":

$$\leq 2\ (\text{hasChild} \sqcap \text{hasFemaleRelative})$$

Verschiedene DSLs unterscheiden sich dadurch, wie viele der Konstruktionsmöglichkeiten von Begriffen/Rollen sie enthalten. Besonders ausdrucksstarke DSLs erlauben auch *zyklische* Definitionen, wie man sie letztlich im semantischen Holismus erwartet. Typischerweise wird angenommen, dass jeder Begriff nur eine Definition hat und keine zyklischen Definitionen auftreten.

Semantisch bieten sich Mengen als Extensionen der Begriffe an. Die Domain wird als nicht abgeschlossen betrachtet (*open-world assumption*).[8] Erweiterungen des formalen Apparates betreffen zunächst epistemische Operatoren um mit der *open-world assumption* umzugehen (d.h. das eigene Nichtwissen zu modellieren etc.).

Ein DSL-System besteht aus einer *Terminologie*, welche die Begriffe definiert, und *Weltwissen* (d.h. wahren, aber nicht notwendig wahren Sachverhalten). Der Benutzer stellt diesbezüglich – ähnlich wie in einer PROLOG-Implementation – Fragen. Geprüft wird dann auf logische Implikation (gemäß den Definitionen) oder, falls Individuenausdrücke wie "Anna" zugelassen werden, Zutreffen des Begriffs. In letzterem Fall wird das Vorliegen einer Instanz wie "Female(Anna)" überprüft. *Konsistenz* liegt vor, wenn jeder Begriff instantiiert werden kann.[9]

Als Beispiel kann ein Auszug des semantischen Felds der Verwandtschaftsbeziehungen dienen:

$$\text{Woman} \equiv \text{Peron} \sqcap \text{Female}$$
$$\text{Man} \equiv \text{Person} \sqcap \neg\text{Female}$$
$$\text{Mother} \equiv \text{Woman} \sqcap \exists\text{hasChild.Person}$$
$$\text{Father} \equiv \text{Man} \sqcap \exists\text{hasChild.Person}$$
$$\text{Parent} \equiv \text{Father} \sqcup \text{Mother}$$
$$\text{Grandmother} \equiv \text{Mother} \sqcap \exists\text{hasChild.Parent}$$
$$\text{MotherWithManyChildren} \equiv \text{Mother} \sqcap \geq 3\ \text{has Child}$$

---

[8] Im Unterschied zu Datenbanken, wo fehlende Information als Verneinung aufgefasst wird und Anfragen entsprechend "nur" die finiten Modelle prüfen, wird in DSL-Systeme fehlende Information als *momentan* nicht vorhandenes Wissen bewertet, sodass die Bewertung von Anfragen entsprechend komplexer wird.

[9] Auch dieser Begriff der Konsistenz ist natürlich nicht der übliche, etwa der Prädikatenlogik.

MotherWithoutDaughter ≡ Mother⊓ ∀hasChild.¬Woman
Wife ≡ Woman ⊓ ∃hasHusband.Man

Basal sind hier nur: Person, Female, hasChild, hasHusband. Eine Expansion der Terminologie löst die rechten Seiten der Definitionen in die basalen Begriffe/Rollen auf. Mittels einer solchen Expansion kann geprüft werden, ob Definitionszyklen vorliegen. Die Expansion kann jedoch exponentiell in der Länge der nichtexpandierten Terminologie sein.

In einer der obigen Terminologie (T-Box) entsprechenden A-Box (für das Weltwissen) könnten sich befinden:

Father(Peter)
hasChild(Peter,Harry)
MotherWithoutDaughters(Mary)
hasChild(Mary,Peter) …

Alle logischen Prüfungen eines DSL-Systems lassen sich reduzieren auf den Konsistenz/Erfüllbarkeitstest für A-Boxen (z.B. gilt A⊑B genau dann, wenn A⊓¬B unerfüllbar ist), wobei die A-Boxen mittels der Expansionen der T-Boxen umformuliert werden. Ein Begriff B ist erfüllbar, wenn die Instanz B(a) konsistent ist, wenn die entsprechende A-Box eine Interpretation hat, die ein Modell für die A-Box und die T-Box liefert. Für einige Systeme lassen sich semantische Tableau-Regeln angeben, die an die entsprechenden Regeln der Standardlogik anknüpfen.

Die Komplexität von DSLs ergibt sich korrespondierend zu ihrer Ausdrucksstärke.

• FL^- enthält Schnitt, Rollen mit Allquantoren und Existenzquantoren, die *nur* das Vorhandensein eines Relatums behaupten (∃R). Negation fehlt. Das Subsumptionsproblem ist in P.

• *FL* erweitert FL^- um Rollenbeschränkungen $R_{|C}$ (die Menge der R-Relate, die auch C sind, ohne dass *alle* R-Relate C sein müssen). Subsumption in *FL* ist co-NP-hart. Auch die Erweiterung von FL^- um allgemeine Existenzquantoren (∃R.C) führt zu einen NP-harten Subsumptionsproblem.

• *AL* enthält Negation bezüglich atomarer Begriffe, Schnitt, Rollen mit Allquantoren und Existenzquantoren, den Universellen Begriff ⊤ sowie den Ausschließenden Begriff ⊥. Subsumptionsprüfung in *AL* ist in P. Erweitert man *AL* um mindestens zwei Rollennamen und Inklusionsaxiome der Form A⊑B ist der Test auf die Unerfüllbarkeit eines Begriffes B EXT-hart. Erlaubt man zyklische Axiome wird *AL*-SAT PSPACE-hart; das bleibt auch dann so, wenn man zusätzlich numerische Quantoren zulässt.

• *ALC* erweitert *AL* um allgemeine Negation. Damit ergibt sich mindestens eine Entsprechung von Begriffen und ihrer Zusammensetzung zur Aussagenlogik. Daher ist *ALC*-SAT mindestens NP-hart. Tatsächlich ergibt sich durch die Quantoren eine Korrespondenz zu QBF, so dass *ALC*-SAT PSPACE-hart ist.

• ALC_{trans} erweitert *ALC* um Vereinigung, Schnitt und transitive Hülle von Rollen. Es handelt sich damit um eine Sprache, die **PDL** korrespondiert (Begriffe korrespondieren Aussagen und Rollen korrespondieren Programmen). Deshalb ist ALC_{trans}-SAT EXT-*vollständig*. Der Erweiterung von **PDL** um inverse Programme entspricht $ALCI_{reg}$. $ALCI_{reg}$ erweitert ALC_{trans} um inverse Rollen, um Rollen/Begriffs-Identität [Id(A)], um allgemeine Inklusionsaxiome der Form $A \sqsubseteq B$ mit komplexen Ausdrücken A und B sowie um zyklische Definitionen. $ALCI_{reg}$-SAT ist EXT-*vollständig*. (Die entscheidende Komponente der Komplexitätssteigerung ist also die transitive Hülle.)

• *ALCN* erweitert *AL* um allgemeine Negation und numerische Quantoren. *ALCN*-SAT mittels Tableaus ist in EXT und EXTSPACE (da eine Interpretation durch Entwicklung des entsprechenden Binärbaums geliefert wird)! Durch Wiederverwendung von Speicher (s.o.) lässt sich allerdings durch eine NTM die Erfüllbarkeit in PSPACE beantworten. Erlaubt man *allgemeine Inklusionsaxiome* steigt die Komplexität. *ALCN*-Konsistenzprüfungen können dann auf exponentielle Pfade im Suchbaum stoßen, sodass *ALCN&Inclusions* nicht mehr in PSPACE, sondern vielmehr EXT-*vollständig* ist!

Einige DSLs, die noch komplexere Konstruktionen zulassen, sind gar nicht entscheidbar.

Die mit *Description Logics* gegebene Modellierung ist also in einigen Bereichen (einigen simplen Logiken), obwohl auch über Individuen geredet wird, nicht von der vollen Komplexität der Prädikatenlogik oder einiger Modallogiken, jedoch ergibt sich die entsprechende Komplexität wieder, sobald solche semantischen Konstruktionen eingeführt werden, die in natürlichen Sprachen vorliegen bzw. zu erwarten sind. Von besonderem Interesse sind hier komplexe Rollen und zyklische Definitionen.

Trotz der hohen theoretischen Komplexität der entsprechenden DSL-Systeme gibt es eine Reihe von erfolgreichen Implementierungen. Ganz allgemein muss man deshalb nach dem Verhältnis der abstrakten Resultate zu tatsächlich auftretenden Berechnungen und sie implementierenden Maschinen fragen, eingeschlossen die semantische Kompetenz und Performanz des menschlichen Geistes/Gehirns.

§5 Gehirnleistungsgrenzen?

Das Suchen durch den Berechnungsbaum einer NTM, welches eine simulierende DTM in die Breite durchführt, hat eine immense Komplexität. Exponentielle Zeit – und Probleme, die in NP liegen lassen sich mutmaßlich nur exponentiell implementieren – führt schnell dazu alle physikalisch verfügbaren Ressourcen zu erschöpfen.

Betrachten wir eine NTM M mit binärer Verzweigung (bzw. den entsprechenden Suchbaum für eine simulierende DTM M*). Der Tiefe der Berechnung entspricht also die Schrittanzahl der Berechung von M – offensichtlich wird hier also eine unrealistisch kurze Berechnung betrachtet, hat man typische TM-Berechnungen vor Augen. Ein gegenwärtiger Computer mag ca. 10.000 Schritte pro Sekunde berechnen, also 10000 Knoten besuchen. Ein Knoten (Zustand, gelesene Zeichen auf Input und Arbeitsbändern) sei ca. 1KB. Das Gehirn (d.h. ein Neuron) rechnet mit maximal ca. 1000 Signalen pro Sekunde. Man kann also eine *sehr grobe* Abschätzung vornehmen:

Tiefe der Berechnung	Knoten im Berechnungs-baum	Berechnungszeit Computer CPU	Berechnungszeit Neuron	Speicherbedarf
10	1000	0.11 Sekunden	1.1 Sekunden	1 MB
16	130000	11 Sekunden	2 Minuten	106 MB
24	10.000.000	19 Minuten	3 Stunden	10 Gigabyte
30	1.000.000.000	31 Stunden	310 Stunden	1 Terrabyte
34	10^{11}	129 Tage	3 Jahre	101 Terrabyte
44	10^{13}	35 Jahre	350 Jahre	10 Petrabyte
50	10^{21}	5932 Jahre	60000 Jahre	1,5 Exabyte

Die Erfüllbarkeit einer Menge von 80 semantisch von einander unabhängigen Elementaraussagen durch eine DTM zu prüfen verlangt so eine Berechnungszeit, welche die Lebenszeit nicht allein jedes Menschen und jeder gebauten Maschine, sondern sogar die des Universums (gemäß gegenwärtiger physikalischer Theorien) übersteigt!

Eine erste philosophische Konsequenz, die man aus diesen Resultaten und Zahlen ziehen könnte, wäre:

(Option 1) Aufgrund von komplexitätstheoretischen Resultaten kann der menschliche Geist – implementiert im Gehirn – nicht perfekt rational sein.

Auf der anderen Seite muss man vorsichtig sein, die absolut groben Abschätzungen der Komplexitätstheorie direkt aus Aussagen über die Leistungsfähigkeiten des Geistes/des Gehirns zu übernehmen:

- Konstante Faktoren werden nicht berücksichtigt, d.h. $10^{21}*n$ wird als linear angesehen! Demgegenüber wäre $10^{-2100}*2^n$ exponentiell, obwohl das Problem selbst bei großen Werten von n praktikabel bleibt!
- Die Größe der Exponenten wird nicht berücksichtig: n^{1000} wäre in P, aber wohl kaum praktikabel. $2^{n/5000}$ wäre demgegenüber in EXT, obwohl es sehr wohl praktikabel sein kann!
- Es werden die schlimmsten Fälle (die längsten Äste) der Zuordnung zugrunde gelegt, obwohl die große Mehrheit der Berechnungen wesentlich schneller sein mag!
- Es werden (bisher) nur Berechnungen durch DTM und NTM betrachtet.

Die erste philosophische Konsequenz, welche komplexitätstheoretische Resultate nahe legen, (Option 1) könnte also zu voreilig sein.

Bezüglich einer solchen Relativierung der komplexitätstheoretischen Resultate stellt sich die Frage, ob es nicht Weisen der Berechnung gibt, bei denen der Aufwand stark vermindert werden kann. Drei weitere Optionen scheinen diskutierbar:

(Option 2) Die vorgesellten Resultate betreffen das Modell der sequentiellen Turing-Maschinen. Das Gehirn arbeitet jedoch massiv parallel. Also müsste eine beachtliche Beschleunigung möglich sein.

(Option 3) Die vorgestellten Algorithmen arbeiten fehlerfrei. Eine beachtliche Beschleunigung sollte möglich sein, wenn man gelegentliche Fehler zugesteht, deren Wahrscheinlichkeit aber immens begrenzt.

(Option 4) Die Komplexitätsmaße ergeben sich aus der Inputlänge. Eine beachtliche Beschleunigung sollte möglich sein, wenn man ein Problem in Teilprobleme zerlegt.

Gehen wir diese Optionen einzeln durch:

(ad Option 2)

Zur Beurteilung dieser Option müssen Modelle paralleler Berechenbarkeit betrachtet werden. Genauso wie es eine Fülle von Modellen sequentieller Berechenbarkeit gibt, gibt es eine Fülle von Modelle von Maschinen, die anknüpfend an die bekannten sequentiellen Maschinen definieren, wie entsprechende parallele Berechnung vonstatten gehen könnte. Zwei Beispiele:

• Eine *Array Processing Machine* (APM) ist eine Registermaschine, die im Unterschied zu einer gewöhnlichen Registermaschine, welche in einem Schritt, der keine Lade-, Null- oder Test/*Jump*-Anweisung ist, jeweils ein Register direkt adressiert, um es zu erhöhen oder herunterzusetzen, nicht *nur* solche sequentiellen Schritte ausführen kann, sondern auch *parallel* solche Anweisungen auf ganze Blöcke von Registern (in gleicher Weise) anwenden kann. Dadurch ergibt sich eine immense Beschleunigung; beispielsweise kann die Summe von n Zahlen in $log(n)$ Schritten berechnet werden!

• Eine *k-PRAM* (*k-offspring parallel random access machine*) als weitere Registermaschinenvariante entspricht ungefähr dem Vorgehen der prozeduralen Programmierung mit verteilter/paralleler Ausführung der Subprozeduren: Es gibt ein Programm, das Subprogramme/Prozeduren enthält, die von eigenen Kontrolleinheiten mit eigenem Speicher ausgeführt und dann zurück in das Hauptprogramm eingespeist werden; insofern gibt es – außer dass es einen Prozessor gibt, der das Programm initiiert – keine zentrale Kontrolle, die das Programm ausführt, und keinen zentralen Speicher (*multiple instruction multiple data stream*). Eine Zeiteinheit ist ein Schritt, der allerdings in vielen parallelen Teilberechnungen erfolgen kann. Die Berechnung entspricht einer Baumstruktur, da die Teilprogramme jeweils ihr Ergebnis an die aufrufende Stelle zurückgeben. Die Zeitkomplexität einer *k-PRAM* entspricht daher der Tiefe dieses Berechnungsbaumes. *k-PRAM*s können von DTMs simuliert werden, indem die Berechnungen hintereinander ausgeführt und die Ergebnisse und Aufrufe in einem Stack verwaltet werden:

> (T15) Eine nichtdeterministische *k-PRAM* M mit P-TIME($f(n)$) kann durch eine DTM M* simuliert werden in SPACE($O(f^3(n))$).

Entscheidend ist die *These der parallelen Rechenzeit*:[10]

> (PCT) Sequentielle Raumkomplexität entspricht innerhalb eines Polynoms paralleler Zeit; P-TIME($f^{O(1)}$(n))=SPACE($f^{O(1)}$(n)).

Beweisidee: Sei M eine NTM, M* eine nichtdeterministische *k-PRAM*, M∈ NSPACE. M* prüft jeweils für ein Paar von Konfigurationen $<C_1,C_2>$ von M, ob C_2 in einem Schritt aus C_1 erreicht wird; wenn nicht, dann verteilt der aufrufende Prozessor an zwei andere Subprozessoren die Aufgabe von der Mitte C_3 zwischen C_1 und C_2 aus entsprechende Überprüfungen vorzunehmen.[11] Der Baum kann, nach Voraussetzung M∈ NSPACE, nur polynom tief sein. Das Raten/Generieren von C_3 kann auch nicht länger dauern. Somit ergibt sich eine maximal quadratische Schrittanzahl/Verzögerung O(f^2) relativ zur Komplexitätsfunktion f von M. Die Verzögerung bleibt also in polynomen Bereich. [Mit zusätzlicher Verzögerung O(f^4) – also immer noch im polynomen Bereich – kann M* von einer *deterministischen k-PRAM* M** simuliert werden, also M auch von M** innerhalb polynomer Verzögerung O(f^6(|n|)) in der Zeit gegeben die sequentielle Raumkomplexitätsklasse O(f(|n|). 6∈ O(1).] ■

Mit diesem Ergebnis können wir Probleme, die in PSPACE liegen, als mutmaßlich praktikable Probleme auffassen! Dies betrifft die hier betrachteten normalen Modallogiken als auch eine große Zahl der Modellierungen von Semantik mittels Description Logics. Eine konkret umsetzbare Implementierung liegt damit zwar noch nicht vor, doch verschiebt sich entscheidend, was als *zu komplex* gilt.

(ad Option 3)

Probabilistische Turing-Maschinen (PTMs) raten nicht den nächsten Schritt, wie eine NTM, sondern jeder Schritt hat eine bestimmte Wahrscheinlichkeit. Die Wahrscheinlichkeit, dass ein Input akzeptiert wird, ist die Summe der Wahrscheinlichkeiten der Pfade, die den Input akzeptieren.

---

[10] Vgl. (Balcazar/Diaz/Gabarro 1990, Kap. 2). Die These geht zurück auf Arbeiten von Chandra und Stockmeyer aus dem Jahr 1976. Es gibt sogar nicht-deterministische parallele Automatenmodelle, die eine noch stärkere Beschleunigung mit sich bringen.

[11] Dieses Vorgehen entspricht dem Simulieren einer NTM M∈ NSPACE durch eine DTM M* in PSPACE, wie es beim Beweis von *Savitchs Theorem* verwendet wird; vgl. (Balcazar/Diaz/Gabarro 1988, S. 50ff., 71f.). Man kann auch sagen: Im Unterschied zu einer simulierenden DTM M', die (schlimmstenfalls) den Binärbaum der NTM M in exponentieller Zeit durchlaufen muss, testet die simluriende *k-PRAM* mögliche Äste einer erfolgreichen Berechnung parallel.

Zur Kanonisierung kann man annehmen, dass immer zwei Optionen offen stehen, dass alle Pfade nur eine bestimmte Schrittlänge lang sind ("*clocked*", wie bei zeitgbegrenzten Universellen Turing-Maschinen) und dass alle Pfade entweder in einem Akzeptanz- oder Zurückweisungszustand enden. Es gibt sich ein Binärbaum mit der Zeitkomplexität als Höhe. Im Gegensatz zu NTMs gilt ein Wort nur dann akzeptiert, wenn nicht allein eine, sondern *die Mehrheit* der Berechnungen akzeptiert. Alle Berechnungen benötigen dieselbe Zeit (bzw. werden entsprechend gestoppt). Darüber hinaus können PTMs Fehler machen: eine Berechnung akzeptiert, obwohl der Input inakzeptabel ist. Die *Fehlerwahrscheinlichkeit* ist das Verhältnis der sich irrenden Berechnungen zur Anzahl aller Berechnungen. Für diese *Fehlerwahrscheinlichkeit* einer Maschine M kann auch eine Grenze e_M festgelegt werden.

PTMs können durch DTMs mit exponentieller Verzögerung simuliert werden:

> (T16) Sei M einer PTM mit PTIME(f(|n|)) dann simuliert die DTM M* M in SPACE(f(|n|)) und TIME($2^{f(|n|)}$).

Die simulierende DTM muss zur Simulation eventuell den ganzen Binärbaum ablaufen, kann aber den beschriebenen Platz wiederverwenden. Auch die Definition des Akzeptierens durch eine PTM ist dabei in einer gewissen Weise „unfair“ gegenüber dem Akzeptieren durch eine DTM. Die einzelnen Berechnungen einer PTM in PP sind zwar polynom in der Zeit, damit ist aber noch nicht klar, woher wir wissen, dass *die Mehrheit* der Äste einen Input akzeptiert hat. Wenn wir warten (also die erfolgreichen Berechnungen *zählen*), dann kann es sein, dass wir eine exponentielle Zeit auf die Antwort, ob eine entsprechende PTM einen Input akzeptiert, warten müssen, denn nichts verbietet – vielmehr ergibt sich dies aus der Definition der kanonisierten PTMs – dass es exponentiell viele Äste im Berechnungsbaum gibt. Warten und Zählen korrespondiert daher dem Simulieren und Zählen, das eine DTM vornehmen muss, um eine PTM zu simulieren!
Es kann jedoch sein, dass wir *a priori* (etwa aufgrund eines Korrektheitsbeweises) wissen, dass eine PTM M tatsächlich mit der Mehrheit ihrer Berechnungen einen Input akzeptiert (bzw. akzeptieren würde, ließen wir sie entsprechend oft laufen).

Zeitkomplexitätsklassen von PTMs ergeben sich relativ zur festgesetzten Fehlerwahrscheinlichkeit. PP ist die Komplexitätsklasse der PTMs mit polynomer Laufzeit und $e_M \leq ½$. Trivialerweise gilt NP$\subseteq$PP. Aufgrund von (T16) gilt also:

> (T17) NP $\subseteq$ PP $\subseteq$ PSPACE.

PP schließt NP *und* co-NP ein, da PP = co-PP. PP enthält also Probleme, die mutmaßlich exponentiell sind, selbst für NTMs.

#SAT ist das Problem, ob die Anzahl der Interpretationen, die eine Formel erfüllen, größer ist als eine mit der Anfrage vorgegebene Zahl *n*.

(T18) #SAT ist PP-*vollständig*.

Die Umstellung auf probabilistische Automaten mit großer Fehlerwahrscheinlichkeit bringt allerdings so eine Beschleunigung, die *höchstens* so stark ist wie die Umstellung auf deterministische parallele Automaten.
Eine DTM, die simuliert, ob eine PTM einen Input akzeptiert, simuliert nicht nur einzelne Äste, sondern den Gesamtdurchlauf durch alle Äste der PTM. Deshalb benötigt die Simulation exponentielle Zeit. Die Prozedur des Berechnens (der Algorithmus) hingegen könnte durch eine höhere Programmiersprache wie PASCAL ausgedrückt werden, indem das Generieren einer Zufallszahl für den nicht-determinierten Schritt verwendet wird; auf der Ebene der Programmbeschreibung bringt dies jeweils nur eine Verzögerung um eine Konstante mit sich; auf der Ebene des kompilierten Programms liegt zwar nicht wirklich Indeterminismus vor, entscheidend ist jedoch die Ebene des Programms selbst. Jeder Durchlauf durch dieses Programm generiert dann einen Ast des PTM-Berechnungsbaums, wobei verschiedene Durchläufe verschiedene Äste realisieren können. Jeder Durchlauf benötigt für eine Maschine, deren Algorithmus eine Menge aus PP erzeugt, polynome Zeit.
Die Beschleunigung, die sich mit PTMs nun anbietet, verfährt wie folgt:
Angenommen wir *wissen*, dass eine PTM M korrekt bezüglich einer Frage ist (d.h. die Mehrheit der Berechnungen würde einen Input akzeptieren). Ein Durchlauf liefert ein Ergebnis mit einer Fehlerwahrscheinlichkeit $x < ½$. Die Durchläufe sind probabilistisch unabhängig von einander. Lässt man also die Maschine/das Programm sehr oft durchlaufen – sagen wir polynom $p(|n|)$ oft in der Länge des Inputs n – dann reduziert sich der Fehler auf $½^{p(|n|)}$! Die Gesamtwartezeit bei polynom vielen Rechnungen polynomer Länge bleibt polynom. Das heißt, dass wir, wenn wir einen gewissen – sei es auch sehr kleinen – Fehler zulassen, recht sicher sein können, dass korrekte Ergebnis vorliegen zu haben, obwohl wir nur polynome Zeit investiert haben.
Wieweit diese Beschleunigung geht, ist indessen nicht so klar: Die entsprechende Komplexitätsklasse BPP liegt nicht oberhalb, sondern neben NP, d.h. es besteht keine Garantie NP-harte Probleme so angehen zu können. BPP liegt in der Anordnung der Komplexitätsklassen *neben* NP und co-NP. Die Klasse ZPP, bei der gar keine Fehler auftreten, sondern höchstens eine Berechnung keine Antwort gibt, liegt sogar innerhalb von NP.

Man kann also ein System bauen, das auf einer Ebene der Beschreibung (etwa eines Algorithmus einer höheren Programmiersprache, welche die Generierung von Zufallszahlen erlaubt) eine PTM ist. In der gerade beschriebenen Weise werden damit Probleme – bis auf einen geringen Fehler – lösbar, die oberhalb von P liegen. Auf der Ebene der letztlichen Implementierung/Compilierung handelt es sich dann aber wieder um eine DTM mit entsprechender Verzögerung betrachtet man das Gesamtsystem und die Frage, ob die *Mehrheit* der Berechungen akzeptiert *hat*.[12] Bezüglich der Semantik wäre zu untersuchen, ob es interessante Modellierungen (etwa mittels spezieller Description Logics) gibt, die in die entsprechende Klasse fallen.

(ad Option 4)

Die Beschleunigung, die sich durch Aufspalten eines Problems ergeben kann, kann massiv sein. 2^{80} Knoten in einem Binärbaum zu besuchen übersteigt alle vorhandenen Ressourcen. 2^{20} ist etwas mehr als 1 Million. Die Aufspaltung des größeren Problems in nur 4 gleichgroße kleinere Probleme erlaubt das praktisch unlösbare Problem in – je nach Maschine – Minuten oder Sekunden zu lösen! Eine Strategie, Komplexitätsgrenzen zu umgehen, kann daher darin liegen, Probleme so weit wie möglich in Teilprobleme aufzuspalten. Statt ganze mögliche Welten und ihre Konsistenz und Konsequenzen zu betrachten, liefern sehr partielle Situationsbeschreibungen mutmaßlich alles, was wir in einer entsprechenden Lage berücksichtigen müssen.

Diese Option der Beschleunigung gekoppelt mit paralleler Berechnung kann so zu erheblicher Beschleunigung führen. Dies schließt in diesem Falle – wie im gerade diskutierten Beispiel – Probleme *exponentieller Härte* ein.

*

---

[12] Ähnliches gilt für eine Reihe in der Künstlichen Intelligenz verwendeten Algorithmen (wie *Baysianischen Netzwerke*). Das Auswerten einer Frage bezüglich einer Wahrscheinlichkeitsverteilung, das alle Abhängigkeiten berücksichtigt, ist – entsprechend der kombinatorischen Explosion bei Wahrheitstafeln – exponentiell. Testen einer Schlussfolgerung in *Baysianischen Netzwerken* ist (deshalb) NP-hart. Es kommt dann bei den vorhandenen erfolgreichen Implementierungen alles darauf an, problembezogene Vereinfachungen eines jeweiligen Algorithmus oder einer Problembeschreibung zu finden, indem etwa einzelne Variablen als konstant oder irrelevant angesehen werden; vgl. (Russell/Norvig 1995, Kap.14 und 20).

Man sollte die Resultate/Theoreme der Komplexitätstheorie somit nicht einfach als negative Resultate bezüglich der Grenzen der (menschlichen) Rationalität interpretieren. Sie liefern vielmehr *constraints*, die eine realistische Theorie der Kognition und damit auch der Semantik berücksichtigen muss.

Insbesondere kann man die entsprechenden Theoreme unter der Annahme, dass wir hinreichend kognitiv/logisch/semantisch kompetent und auch performant sind, als Belege für spezifische Modelle der Kognition ansehen, die Probleme, die in naiven DTM-Lösungen zu hart sind, für uns lösbar machen.

Literatur

• Einen Überblick über die Grundlagen Komplexitätstheorie liefert (Balcazar/Diaz/Gabarro 1988), der zweite Band (Balcazar/Diaz/Gabarro 1990) behandelt weitergehende Themen wie parallele Berechenbarkeit. Einen knappen Überblick im Rahmen einer Theorie auch der formalen Sprachen liefert (Hopcroft/Ullman 1979). Zur Berechenbarkeit im Allgemeinen siehe (Cutland 1980).

• Zur Metatheorie der Modallogik im Allgemeinen und deren Komplexität vgl. (Blackburn/de Rijke/Venema 2001).

• Zur Description Logic im Allgemeinen und deren Komplexität vgl. (Baader et al. 2003). Anwendungsgebiete der DSL neben sprachlicher Modellierung im engeren Sinne sind etwa Softwaremodellierung und Medizininformatik.

• Zu psychologischen Untersuchungen der Performanz beim Räsonieren und entsprechenden Tests siehe (Manktelow 1999).

• Zu genaueren Einschätzungen der Komplexität verschiedener anwendungsbezogener Algorithmen und deren Rolle in der Künstlichen Intelligenz siehe (Russell/Norvig 1995/2003). Dort werden auch eine Reihe von Verfahren behandelt, hochkomplexe Probleme durch Annäherung oder problembezogene Vereinfachung praktikabel zu machen.

© Manuel Bremer 2004/2005.