(12) a. Boris ljubit Tanju. [Russisch]
Boris loves Tanja
b. **Kto** ljubit Tanju?
who loves Tanja
c. **Kogo** Boris ljubit? / **Kogo** ljubit Boris?
whom Boris loves
d. **Kto kogo** ljubit?
who whom loves

Einige Daten, mit denen wir uns beschäftigen werden:

Fragepronomen sind in vielen Sprachen mit Indefinitpronomina verwandt, auch im Deutschen:

(13) Karl hat was gesehen.

Fragepronomina unterliegen Insel-Beschränkungen:

(14) a. Maria hat wessen Bruder geheiratet?
b. *Wessen hat Maria Bruder geheiratet?
c. Wessen Bruder hat Maria geheiratet?

(15) Maria hat wessen Bruder geheiratet?
*Peter. / *Peters. / Peters Bruder.

1.4 Diskurs-Perspektiven

Fragen haben eine grosse Bedeutung für die Textanalyse. Gut gebaute Texte sind durch implizite Fragen strukturiert, die der Text nach und nach beantwortet. Auf diese Weise kann man die Akzentverhältnisse eines Textes und den Einsatz von

(16) [Was war los?] Wir sind eben von Itálien zurückgekommen. [Wie lang ward ihr dort?] Wir waren da für zwei Wóchen. [Warum ward ihr dort?] Wir wollten uns mal so richtig erhólen. [Wo ward ihr genau?] Zuèrst waren wir in der Toskána, und dànn sind wir noch nach Rímini gefahren.

1.5 Aufgabe

Lesen Sie den Überblicksartikel von Bäuerle & Zimmermann 1993.

1.6 Überblicksartikel zur Semantik der Frage

R. Bäuerle & Th. E. Zimmermann: "Fragesätze". In D. Wunderlich & A. von Stechow (Hrsg.), *Handbuch Semantik*. Walter de Gruyter 1993, 333-348

J. Ginzburg: "Interrogatives. Questions, facts and dialogues". In S. Lappin (ed.), *Handbook of Contemporary Semantic Theory*. Blackwell 1995, 385-422.

J. Groenendijk & M. Stokhof: "Questions". In J. van Benthem & A. ter Meulen (eds.), *Handbook of Logic & Languages*. Elsevier 1998, 1055-1124.

2. Überblick: Wahrheitsfunktionale Semantik von Aussagen

Ziel: Knappe Einführung in die Grundannahmen und –Notationen, wie wir sie für die Semantik zugrundelegen. Dies kann eine systematische Einführung nicht ersetzen (z.B. Grundkurs C Semantik (siehe Website M. Krifka); Heim & Kratzer, *Semantics in Generative Grammar*; Chierchia & McConnell-Ginet, *Meaning and Grammar*

2.1 Die Bedeutung von Aussagesätzen

wird auf den Begriff der **Wahrheit** zurückgeführt. Die Bedeutung eines Satzes Φ (für die wir **[**Φ**]** schreiben) erlaubt es uns, unter jedem Umstand i zu sagen, ob Φ **wahr** ist oder **falsch**., d.h. seinen **Wahrheitswert** anzugeben. (Über etwaige zusätzliche Bedeutungskomponenten können wir hinwegabstrahieren)

Wir nennen solche 'Umstände' auch **mögliche Welten** (d.h. mögliche Zustände der Welt). Es ist vorteilhaft, **[**Φ**]** mit der Menge der möglichen Welten zu identifizieren, in denen Φ wahr ist. Wir nennen solche Mengen **Propositionen**. Wir verwenden i, i′, j usw. als Variable für mögliche Welten, I als Menge aller möglichen Welten.

(17) [*Berlin ist die Hauptstadt von Deutschland*] =
{i | Berlin ist die Hauptstadt von Deutschland in i}

Der Begriff der möglichen Welt ist auch temporal zu verstehen (d.h. Welt zu einem Zeitpunkt). Wir schreiben i < i′ wenn i zeitlich vor i′ liegt.

(18) [*Bonn war die Hauptstadt von Deutschland*] =
{i | Es gibt ein i′ mit i′<i, sodass gilt: Bonn ist die Hauptstadt von Deutschland in i′}

Die Interpretation von Aussagesätzen durch Mengen möglicher Welten erlaubt es, die Bedeutung von Verbindungen von Aussagesätzen wie folgt zu berechnen:

(19) [Φ *und* Ψ] = [Φ] ∩ [Ψ]
[Φ *oder* Ψ] = [Φ] ∪ [Ψ]
[*Nicht* Φ] = I – [Φ] Beispiel: Venn-Diagramme!

Die nächste Frage ist: Welche Bedeutung haben die Konstituenten und Wörter unterhalb der Satzebene? Hierzu müssen wir erst einige Notationsweisen für Funktionen kennenlernen.

2.2 Funktionen und Lambda-Ausdrücke

Eine **Funktion** ist eine **Zuweisungsvorschrift**, die jedem Element aus einer Menge von **Argumenten** (dem **Argumentbereich** oder der **Domäne**) einen **Wert** zuweist. Beispiele:

(20) a. f(x) = 2x + 1, f: 1 → 3, 2 → 5, 3 → 7, 4 → 9, usw.
b. Die 'Vater'-Funktion: Joseph → Jakob, Jakob → Isaak, Isaak → Abraham, usw.

Funktionen weisen jedem Element des Bereichs genau einen Wert zu (d.h. die 'Sohn'-Funktion ist nicht definiert; 'Sohn' ist vielmehr eine sogenannte **Relation**). Wir sagen, dass wir eine Funktion auf ein Argument **anwenden** oder **applizieren** und dabei einen Wert erhalten. Schreibweise: **Funktion(Argument) = Wert.**

(21) a. f(3) = 7
b. Vater(Jakob) = Isaak.

Wir beschreiben Funktionen im allgemeinen mit Lambda-Termen:

(22) a. $\lambda x[2x + 1]$ c. $\lambda x[x^2 + 2x + 1]$ e. $\lambda x[13]$
b. λx[der Vater von x] d. λx[die Mutter des Vaters von x] f. $\lambda x[x]$

- λx [……x…..]: x ist die **Variable**, die der λ-Term **bindet**,
- […x….] ist die Beschreibung des Wertes der Funktion, in der typischerweise x ein- oder mehrmals vorkommt.
- Wir können den Argumentbereich explizit machen, durch die folgende Schreibweise: $\lambda x \in A[\ldots x \ldots]$, wobei A der Argumentbereich ist. Beispiele:

(23) a. $\lambda x \in N[2x + 1]$ (N: Menge der ganzen Zahlen)
b. $\lambda x \in \{y \mid y$ ist weiblich$\}$[der Vater von x]

Wenn wir eine Funktion in λ-Schreibweise auf ein Argument anwenden, erhalten wir den Wert, indem wir die durch λ gebundene Variable in der Wertbeschreibung ersetzen (**λ-Konversion**):

(24) a. $\lambda x[2 \cdot x + 1](3) = 2 \cdot 3 + 1 = 6 + 1 = 7$
b. $\lambda x[x^2 + 2x + 1](3) = 3^2 + 2 \cdot 3 + 1 = 9 + 6 + 1 = 16$
c. λx[der Vater von x](Jakob) = der Vater von Jakob = Isaak
d. $\lambda x[13](3) = 13$
f. $\lambda x[x](3) = 3$

Der Wert einer Funktion kann selbst wieder eine Funktion sein:

(25) a. $\lambda x[\lambda y[2x + y]]$, kürzer: $\lambda x\lambda y[2x + y]$
b. $\lambda x\lambda y[2x + y](3) = \lambda y[2 \cdot 3 + y] = \lambda y[6 + y]$
c. $\lambda y[6 + y](4) = 6 + 4 = 10$
d. kurz: $\lambda x\lambda y[2x + y](3)(4) = 10$; beachte: $\lambda x\lambda y[2x + y](4)(3) = 11$!

Aber auch die Argumente einer Funktion können selbst wiederum Funktionen sein. Wir verwenden für die Variablen solcher Argumente Großbuchstaben.

(26) a. $\lambda F[F(3)](\lambda x[2 \cdot x + 1])$
$= \lambda x[2 \cdot x + 1](3)$
$= 2 \cdot 3 + 1$
$= 7$

b. $\lambda F[F(3) + F(4)](\lambda x[2 \cdot x + 1]$
$= \lambda x[2 \cdot x + 1](3) + \lambda x[2 \cdot x + 1](4)$
$= 2 \cdot 3 + 1 + 2 \cdot 4 + 1$
$= 7 + 9$
$= 16$

Die Berechnung von Funktionswerten ist in der Regel einfach, eine Sache des Durchhaltevermögens. Eine Schwierigkeit kann sich ergeben, wenn dieselbe Variable mehrfach verwendet wird, wie im folgenden Beispiel, mit angegebenen Bindungsrelationen:

(27) $\lambda x[\lambda F\lambda x[F(3) + x](\lambda y[x + y])(4)](5)$

Wenn wir einfach λ-konvertieren, gerät das letzte x in den Bindungsbereich des zweiten λ-Operators. Abhilfe: Wir wählen eine unterschiedliche Variable (z).

(28) a. $\lambda \underline{x}[\lambda F\lambda \mathbf{x}[F(3) + \mathbf{x}](\lambda y[\underline{x} + y])(4)](5)$
$= \lambda x[\lambda \mathbf{x}[\lambda y[\mathbf{x} + y](3) + \mathbf{x}](4)](5)$ (!!!)
$= \lambda x[\lambda x[x + 3 + x](4)](5)$
$= \lambda x[11](5)$
$= 11$

b. $\lambda x[\lambda F\lambda \mathbf{z}[F(3) + \mathbf{z}](\lambda y[x + y])(4)](5)$
$= \lambda x[\lambda z[\lambda y[x + y](3) + z](4)](5)$
$= \lambda x[\lambda z[x + 3 + z](4)](5)$
$= \lambda x[x + 3 + 4](5)$
$= 5 + 3 + 4$
$= 12$

2.3 Mengen und charakteristische Funktionen

Unter einer **charakteristischen Funktion** wird eine Funktion von Elementen in die Menge der Wahrheitswerte {0, 1} verstanden. Nehmen wir an, dass Beschreibungen wir $[3 \leq 7]$, $[7 \leq 3]$, [Isaak ist eine Frau] usw. für die Wahrheitswerte stehen (hier 1, 0, 0). Dann können wir charakteristische Funktionen mit λ-Ausdrücken darstellen:

(29) a. $\lambda x[x \leq 3]$: $1 \rightarrow 1$, $2 \rightarrow 1$, $3 \rightarrow 1$, $4 \rightarrow 0$, $5 \rightarrow 0$, usw.
b. λx[x ist eine Frau]: Isaak→0, Abraham→0, Rebekka→1, Rachel→1, Joseph→0, …

Charakteristische Funktionen sind äquivalent zu **Mengen:** Jeder Menge M aus einem Grundbereich U (dem Universum) entspricht eine charakteristische Funktion χ_M:

(30) χ_M = diejenige Funktion f von U in {w, 0} sodass gilt:
für alle $x \in U$, $f(x) = w$ gdw. $x \in M$

Beispiel: $\lambda x[x \leq 3] \sim \{1, 2, 3\}$, $\lambda x[3 \leq x \leq 8] = \{3, 4, 5, 6, 7, 8\}$.

Wir können damit Propositionen als λ-Ausdrücke darstellen:

(31) [*die Hauptstadt von Deutschland ist Berlin*]
= λi[die Hauptstadt von Deutschland ist Berlin in i]

Angewendet auf eine bestimmte Welt i_0 erhalten wir einen Wahrheitswert (1 falls in i_0 Berlin die Hauptstadt von Deutschland ist, sonst 0).

2.4 Kompositionalitätsprinzip

Wir haben komplexe Sätze **kompositional** interpretiert, d.h. ihre Bedeutung auf die Bedeutung der Teilsätze zurückgeführt. Dies ist eine zentrale These der Semantik (Frege):

> Die Bedeutung eines komplexen Ausdrucks ist abhängig von der Bedeutung seiner unmittelbaren syntaktischen Teile und der Art und Weise, wie sie zusammengefügt sind.

Das Kompositionalitätsprinzip erlaubt es zu verstehen, wie man mit der Sprache "mit endlichen Mitteln unendlichen Gebrauch" machen kann (W. von Humboldt). Endlich viele Wörter und syntaktische Konstruktionen können verwendet werden, um unendlich viele, auch neue, Gedanken auszudrücken.

2.5 Ausdrücke unterhalb der Satzebene

Bisher betrachteten wir die Bedeutung von Ausdrücken oberhalb der Satzebene. Wir können die Bedeutungen von subsententialen Konstituenten erschließen, wenn wir dem Kompositionalitätsprinzip folgen.

(32) a. [*Molly schläft*] = λi[Molly schläft in i], die Proposition dass Molly schläft
b. [*Molly*] = Molly, die konkrete Person mit Namen Molly
c. [*schläft*] = $\lambda x\lambda i$[x schläft in i],
die Funktion, die jedem Element x die Proposition zuordnet, daß Molly schläft.

(33) [*Molly schläft*] = [*schläft*]([*Molly*])
= $\lambda x\lambda i$[x schläft in i](Molly)
= (i[Molly schläft in i]

Es gibt Gründe, die Weltvariable i jeweils als erste anzugeben; dies erlaubt, zwischen der Intension und der Extension eines Ausdrucks zu unterscheiden. Die Kombination von Bedeutungen muss darauf Rücksicht nehmen.

(34) $[\![\alpha\ \beta]\!]$ = $\lambda i[[\![\alpha]\!](i)([\![\beta]\!](i))]$ oder $\lambda i[[\![\beta]\!](i)([\![\alpha]\!](i)]$ oder
$\lambda i[[\![\alpha]\!]([\![\beta]\!](i))]$ oder $\lambda i[[\![\beta]\!](i)([\![\alpha]\!]]$ oder
$\lambda i[[\![\alpha]\!](i)([\![\beta]\!])]$ oder $\lambda i[[\![\beta]\!]([\![\alpha]\!](i)]$ oder (je nachdem,
$\lambda i[[\![\alpha]\!]([\![\beta]\!])]$ oder $\lambda i[[\![\beta]\!]([\![\alpha]\!]]$ was wohlgeformt ist)

(35) [[*Molly schläft*]] = λi[[[*schläft*]](i)([[*Molly*]])]
= λi[λi′λx[x schläft in i′](i)(Molly)]
= λi[λx[x schläft in i](Molly)]
= λi[Molly schläft in i]

Darstellung von zweistelligen Verben, VP-Negation, Präteritum:

(36) a. [*liebt*] = λiλyλx[x liebt y in i]
b. [*Leopold liebt Molly*]
= λi[[*liebt Molly*](i)([*Leopold*])]
= λi[λi′[[[*liebt*]](i′)([[*Molly*]])](i)([[*Leopold*]])]
= λi[[*liebt*](i)([*Molly*])([*Leopold*])
= λiλyλx[x liebt y in i](Molly)(Leopold)
= [Leopold liebt Molly in i]

(37) a. [*nicht*] = λRλx[nicht R(x)]
b. [*Molly schläft nicht*]
= λi[[*schläft nicht*](i)([*Molly*])]
= λi[[*nicht*]([*schläft*](i))([*Molly*])]
= λi[nicht R(x)](λy[y schläft in i])(Molly)
= λi[nicht Molly schläft in i]

(38) a. [*PRÄT*] = λiλPλx[Es gibt ein i′ < i: P(x)(i)]
b. [*schlief*] = λi[[*PRÄT*](i)([*schlaf-*](i)]
= λiλx[Es gibt ein i′ < i: [*schlaf-*](x)(i)]

2.6 Typen

Die Komplexität der Bedeutung eines Ausdrucks kann durch seinen **Typ** eingeschätzt werden.

(39) a. Grundtypen: t (Wahrheitswerte), e (Entitäten), s (Indizes, mögliche Welten)
b. Wenn σ, τ Typen sind, dann ist (σ)τ ein Typ (von Funktionen von σ-Entitäten in τ-Entitäten). Wenn σ ein Grundtyp ist, werden die Klammern weggelassen.

(40) a. Molly: e
b. [Molly schläft in i]: t
c. λi[Molly schläft in i]: st
d. λxλi[x schläft in i]: est
e. λyλxλi[x kennt y in i]: eest
f. λPλxλi[nicht P(x)(i)]: (est)est

2.7 Aufgaben

- $\lambda F\lambda x\lambda y[F(x)(y)](\lambda u\lambda v[2u + v])(7)(12) = ?$
- Gib die Bedeutungen von *und* als VP-Konjunktion an und berechne die Bedeutung von: *Molly* [*schläft und schnarcht*].
- Gib die Bedeutung von *tief* als VP-Modifikator an und berechne die Bedeutung von: *Molly* [*schläft tief*].
- Gib die Typen der folgenden Ausdrücke an:
-- *und* als VP-Konjunktion (*schläft und schnarcht*)
-- *sehr* als Modifikator eines VP-Modifikators (*schläft* [*sehr tief*])
-- *und* als Konjunktion für transitive Verben (*Leopold* [[*kennt und schätzt*] *Molly*])

2.8 Weitere Konventionen

Wir verwenden eine Reihe von Konventionen für die Beschreibung von Bedeutungen, die wir aus den formalen Sprachen der Logik entlehnen.

Wahrheitswertoperatoren:

- **Negation**: ¬, eine Funktion, einen Wahrheitswert auf einen Wahrheitswert abbildet, und zwar nach der folgenden Vorschrift: 1 → 0, 0 → 1
- **Konjunktion**: ∧, eine Funktion, die zwei Wahrheitswerte auf einen abbildet, und zwar nach der folgenden Vorschrift: (1,1) → 1 und (1,0), (0,1), (0,0) → 0.
- **Disjunktion**: ∨, Vorschrift: (0,0) → 0 und (1,1), (1,0), (0,1) → 1
- **Konditional**: →, Vorschrift: (1,0) → 0 und (1,1), (0,1), (0,0) → 1
- **Bikonditional**: ↔, Vorschrift: (1,1), (0,0) → 1 und (1,0), (0,1) → 0

Die Bedeutung der Konjunktion, Disjunktion und Negation kann wie folgt dargestellt werden:

(41) a. [*und*] = λpλq[p ∧ q]
[*Es regnet und es schneit*]
= λi[[*und*]([*es regnet*](i)([*es schneit*](i))]
= λi[[*es regnet*](i) und [*es schneit*](i)]
= λi[λj[es regnet in j](i) und λj[es schneit in j](i)]
= λi[es regnet in i und es schneit in i]
1. [*oder*] = λpλq[p ∨ q]
2. [*Es ist nicht der Fall dass*] = λp[¬ p(i)]

Wir können die VP-Negation (37.a) und eine Konjunktion für VPn wie folgt definieren:

(42) a. [*nicht*] = λPλx[¬P(x)]
b. [*und*] = λP′λPλx[P′(x) ∧ P(x)]
[*schläft und schnarcht*] = λi[[*und*]([*schläft*](i))([*schnarcht*](i))]
= λiλP′λPλx[P′(x) ∧ P(x)]([*schläft*](i))([*schnarcht*](i))
= λxλi[[*schläft*](x)(i) ∧ [*schnarcht*](x)(i)]

Existenz- und Universalquantor:

- ∃x[…x…]: Es gibt ein x, dass die Bedingung […x…] erfüllt.
- ∀x[…x…]: Alle x erfüllen die Bedingung […x…].

Wir können damit Propositionen der folgenden Art definieren:

(43) a. λi∃x[[*schläft*](x)(i)]: ‘Jemand / etwas schläft.’
b. λi∀x[[*schläft*](x)(i)]: ‘Alle / alles schläft.’

Mithilfe der Wahrheitswertoperatoren können wir die folgenden Propositionen ausdrücken:

(44) a. λi∃x[[*Person*](x)(i) ∧ [*schläft*](x)(i)]: ‘Jemand (= eine Person) schläft.’
b. λi¬∃x[[*Person*](x)(i) ∧ [*schläft*](x)(i)]: ‘Niemand (= keine Person) schläft.’
c. λi∀x[[*Person*](x)(i) ∧ [*schläft*](x)(i)]: ‘Alles ist eine Person, die schläft.’
d. λi∀x[[*Person*](x)(i) → [*schläft*](x)(i)]:
‘Für alles gilt: wenn es eine Person ist, schläft es.’, d.h. ‘Jede Person schläft.’

2.9 Nominalphrasen

Bisher haben wir lediglich Namen wie *Molly* als Argumente von Verben angenommen. Wir finden jedoch auch sogenannte **Quantoren**:

(45) *Jeder / die meisten / einige / jemand / niemand schlief(en).*

Quantoren können nicht, wie Namen, vom Typ e sein und auf Entitäten referieren (worauf sollte *niemand* referieren?). Sie werden vielmehr als Prädikate vom Typ (est)st analysiert:

(46) $\llbracket \textit{jemand} \rrbracket = \lambda i\lambda P\exists x[\llbracket \textit{Person}\rrbracket(x)(i) \wedge P(x)]$
$\llbracket \textit{jemand schläft} \rrbracket = \lambda i[\llbracket \textit{jemand}\rrbracket(\llbracket \textit{schläft}\rrbracket)$
$= \lambda P\lambda i\exists x[\llbracket \textit{Person}\rrbracket(x)(i) \wedge P(x)](\llbracket \textit{schläft}\rrbracket(i))$
$= \lambda i\exists x[\llbracket \textit{Person}\rrbracket(x)(i) \wedge \llbracket \textit{schläft}\rrbracket(x)(i)]$

(47) $\llbracket \textit{jeder} \rrbracket = \lambda i\lambda P\forall x[\llbracket \textit{Person}\rrbracket(x)(i) \rightarrow P(x)]$
$\llbracket \textit{jeder schläft} \rrbracket = \lambda i\lambda x[\text{x ist eine Person in i} \rightarrow \text{x schläft in i}]$

Wir können Quantoren auch in mengentheoretischer Schreibweise darstellen. Dies erlaubt es us, die Beschreibung auf komplexere Quantoren auszudehnen.

Bedeutung anderer Quantoren:

(48) b. $\llbracket \textit{jeder} \rrbracket = \lambda i\lambda P[\{x|\llbracket \textit{Person}\rrbracket(x)(i)\} \subseteq \{x \mid P(x)\}]$
c. $\llbracket \textit{jemand} \rrbracket = \lambda i\lambda P[\{x|\llbracket \textit{Person}\rrbracket(x)(i)\} \cap \{x \mid P(x)\} \neq \emptyset]$
d. $\llbracket \textit{niemand} \rrbracket = \lambda i\lambda P[\{x|\llbracket \textit{Person}\rrbracket(x)(i)\} \cap \{x \mid P(x)\} = \emptyset]$
e. $\llbracket \textit{einige} \rrbracket = \lambda i\lambda P[\#(\{x|\llbracket \textit{Person}\rrbracket(x)(i)\} \cap \{x \mid P(x)\}) > 2]$
f. $\llbracket \textit{die meisten} \rrbracket = \lambda i\lambda P[\#(\{x|\llbracket \textit{Person}\rrbracket(x)(i)\} \cap \{x \mid P(x)\}) > 1/2\ \#\{x|\llbracket \textit{Person}\rrbracket(x)(i)\}]$

Mit dieser Analyse ist die Behandlung von Nominalphrasen wie *jeder Student, die meisten Studenten, einige Studenten, kein Student* ein leichtes. Ein Determinator wie *jeder* nimmt zuerst ein Nomenargument (Typ est) und liefert einen Quantor (Typ (est)st):

(49) $\llbracket \textit{jeder}_D \rrbracket = \lambda P'\lambda P\forall x[P'(x) \rightarrow P(x)]$
$\llbracket \textit{jeder}_D\ \textit{Student} \rrbracket = \lambda i[\llbracket \textit{jeder}_D\rrbracket\ (\llbracket \textit{Student}\rrbracket(i))]$
$= \lambda i\lambda P'\lambda P\forall x[P'(x)(i) \rightarrow P(x)](\llbracket \textit{Student}\rrbracket)$
$= \lambda i\lambda P\forall x[\llbracket \textit{Student}\rrbracket(x)(i) \rightarrow P(x)]$
$\llbracket \textit{jeder}_D\ \textit{Student schläft} \rrbracket = \lambda i[\llbracket \textit{jeder Student}\rrbracket(i)(\llbracket \textit{schläft}\rrbracket(i))]$
$= \lambda i\lambda P\forall x[\llbracket \textit{Student}\rrbracket(x)(i) \rightarrow P(x)](\llbracket \textit{schläft}\rrbracket)]$
$= \lambda i\lambda P\forall x[\llbracket \textit{Student}\rrbracket(x)(i) \rightarrow \llbracket \textit{schläft}\rrbracket(x)(i)]$

2.10 Syntaktische Bewegungen und Quantoren in Objektsposition

Syntaktische Konstituenten können manchmal an nicht-kanonischer Stelle erscheinen. Es wird in der Regel angenommen, dass solche Ausdrücke unter Hinterlassung einer **Spur** (trace) bewegt wurden:

(50) a. *Molly, Leopold loves* _: *Molly*$_1$, *Leopold loves* t_1.
b. *der Student, den Leopold* _ *traf*: *der Student den*$_1$ *Leopold* t_1 *traf.*

Der semantische Effekt von Bewegungen ist die Formation eines Lambda-Ausdrucks:

(51) a. $\llbracket \textit{Leopold loves}\ t_1 \rrbracket = \lambda i\lambda x_1[\text{Leopold liebt } x_1 \text{ in i}]$
b. $\llbracket \textit{Leopold}\ t_1\ \textit{traf} \rrbracket = \lambda i\lambda x_1[\text{Leopold traf } x_1 \text{ in i}]$

Diese Funktionen gehen dann in die Gesamtbedeutung ein. Dies ist relativ einfach im Fall der Topikalisierung (50.a) (die folgende Analyse berücksichtigt nicht den eigentlichen pragmatischen Effekt der Topikalisierung):

(52) $\llbracket \textit{Molly}_1, \textit{Leopold loves}\ t_1 \rrbracket = \lambda i[\llbracket \textit{Leopold loves}\ t_1\rrbracket(i)(\llbracket \textit{Molly}\rrbracket)]$
$= \lambda i[\lambda x_1[\text{Leopold liebt } x_1 \text{ in i}](\text{Molly})]$
$= \lambda i[\text{Leopold liebt Molly in i}]$

In dieser Repräsentation ist noch nicht klar, wie die Bedeutung von *Leopold loves* t_1 kompositionell abgeleitet wird. Dafür gibt es verschiedene Verfahren. Wir werden im allgemeinen wie folgt vorgehen (cf. Heim & Kratzer 1998):

- die Spur führt eine Variable mit demselben Index ein;
- wir nehmen eine syntaktische Operation an, die diese Variable durch einen Lambda-Operator bindet.

(53) a. $\llbracket \textit{loves} \rrbracket = \lambda i\lambda y\lambda x[\text{x liebt y in i}]$
b. $\llbracket t_1 \rrbracket = x_1$
c. $\llbracket \textit{loves}\ t_1 \rrbracket = \lambda i[\llbracket \textit{loves}\rrbracket(i)(\llbracket t_1\rrbracket] = \lambda i\lambda x[\text{x liebt } x_1 \text{ in i}]$
d. $\llbracket \textit{Leopold loves}\ t_1 \rrbracket = \lambda i[\llbracket \textit{loves}\ t_1\rrbracket(i)(\llbracket \textit{Leopold}\rrbracket)] = \lambda i[\text{Leopold liebt } x_1 \text{ in i}]$
e. $\llbracket \lambda t_1[\textit{Leopold loves}\ t_1] \rrbracket = \lambda i\lambda x_1[\text{Leopold liebt } x_1 \text{ in i}]$

Der letzte Schritt erfolgt durch die folgende syntaktische Regel, die einen Bezug zwischen dem Index der Spur und dem Index einer Variablen in der Lambda-Abstraktion herstellt:

(54) $\llbracket \lambda t_n\ \alpha \rrbracket = \lambda i\lambda x_n[\llbracket \alpha\rrbracket(i)]$

Anstelle einer Koindizierung mit einer bewegten Konstitutente, cf. (52), gehen wir also nach dem folgenden Schema vor:

(55) Technische Implementation der Bewegung: $[\ldots\alpha\ldots] \rightarrow \alpha\ \lambda t_n[\ldots t_n\ldots]$

(56) $[_S\textit{Leopold}\ [_{VP}\ \textit{loves Molly}]] \rightarrow [_S\textit{Molly}\ \lambda t_1[_S\textit{Leopold}\ [_{VP}\ \textit{loves}\ t_1]]]$

(57) $\llbracket \textit{Molly}\ \lambda t_1\ [\textit{Leopold loves}\ t_1] \rrbracket$
$= \lambda i[\llbracket \lambda t_1[\textit{Leopold loves}\ t_1]\rrbracket(i)(\llbracket \textit{Molly}\rrbracket)]$

Der Mechanismus der Bewegung kann herangezogen werden, um die Bedeutung von Ausdrücken mit Quantoren in Objektsposition zu beschreiben. Wir nehmen an, dass Quantoren auf der Ebene der Logischen Form (LF) bewegt werden:

(58) *Leopold traf jemanden* $\rightarrow$ *jemanden* $\lambda t_1[\textit{Leopold traf}\ t_1]$
$\llbracket \textit{jemanden}\ \lambda t_1[\textit{Leopold traf}\ t_1] \rrbracket$
$= \lambda i[\llbracket \textit{jemanden}\rrbracket(i)(\llbracket\ \lambda t_1[\textit{Leopold traf}\ t_1]\rrbracket(i))]$
$= \lambda i\lambda P\exists x[\text{x ist eine Person in i} \wedge P(x)](\llbracket\ \lambda t_1[\textit{Leopold traf}\ t_1]\rrbracket(i))]$
$= \lambda i\exists x[\text{x ist eine Person in i} \wedge \llbracket \lambda t_1[\textit{Leopold}\ t_1\ \textit{traf}\ t_1]\rrbracket(i)(x)]$
$= \lambda i\exists x[\text{x ist eine Person in i} \wedge \text{Leopold traf x in i}\}$