Hilfszettel zur Klausur
von Tim S., Seite 1 von 2

1 Fundamental Concepts

Reference: How does the mapping between form and meaning work? Does it work at all?
Compositionality: How are complex utterances built from smaller units? Are they built from smaller units at all?
Combinatorial structure: a small number of meaningless building blocks (phonemes, parts of syllables) combined into an unlimited set of utterances (words and morphemes)
Compositional structure: meaningful building blocks (words and morphemes) are combined into larger meaningful utterances (phrases and sentences)

2 Information Theory

Information Content (Surprisal)

$-log_2 p(x)$: the more frequent the word, the lower its information content. e.g. the word type "blue" occurs ca. 3750 times in 10000 tokens, and its information content is $-log_2(3750/10000)$ ≈1.42 bits.

Shannon Entropy

$H(X) = -\sum p(x) log_2 p(x)$: entropy as probability, the average information content of information encoding units in the language. Measure of information encoding potential of a symbol system. The higher the uncertainty, the larger the entropy. e.g. $H_{char}(Morse) = -(\frac{86}{136} * log_2(\frac{86}{136}) + \frac{50}{136} * log_2(\frac{50}{136})) \approx 0.949$ bits per character.

Joint Entropy, Conditional Entropy

Joint Entropy: $H(X,Y) = -\sum\sum p(x,y) log_2 p(x,y)$
Conditional Entropy: $H(Y|X) = -\sum p(x) \sum p(y|x) log_2 p(y|x)$ The more ambiguity in language (uncertainty), the higher conditional entropy. No ambiguity -> 0.

Probability Estimation

Maximum Likelihood (ML)
Problems: unit problem, sample size problem, interdependence problem, extrapolation problem
Methods: frequency-based, language models, experiments with humans

Mutual Information

$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$
reduction in the uncertainty of X given Y. compromise between minimum learning cost H(Y) and maximum expliciteness I(X; Y).
Entropy is the upper bound on the mutual information between forms and meanings
Is the entropy rate zero? -> asymptotic determinism of human utterances.

3 Propositional Logic

Why formal logic? overcome ambiguity, determine relationships between meanings of sentences, determine meanings of setences, model compositionality, recursive system.

Definition

Proposition: The meaning of a simple declarative sentence
Extensions: real-world situations they refer to.
Frege's Generalization: The extension of a sentence S is its truth value
The proposition expressed by a sentence is the set of possible cases [situations] of which that sentence is true.
propositional variables: p, q, r...
propositional operators: $\neg, conjunction \wedge, disjunction \vee, XOR, \rightarrow, \leftrightarrow$

Syntax: Recursive Definition

(i) Propositional letters in the vocabulary of L are formulas in L.
(ii) If ϕ is a formula in L, then $\neg\phi$ is too.
(iii) If ϕ and ψ are formulas in L, then $(\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$ are too.
(iv) Only that which can be generated by the clauses (i)-(iii) in a finite number of steps is a formula in L.
invalid: $\neg(\neg\neg p), \neg((p \wedge q))$

Construction Trees

$(\neg(p \vee q) \rightarrow \neg\neg q) \leftrightarrow r\ (iii. \leftrightarrow)$
$(\neg(p \vee q) \rightarrow \neg\neg q)\ (iii. \rightarrow)$ $r(i)$
$\neg(p \vee q)\ (ii)$ $\neg\neg q$ (ii)
$p \vee q\ (iii.\vee)$ $\neg q$ (ii)
p (i) q (i) q (i)

Valuation Functions

For every valuation V and for all formulas ϕ: $V(\phi \leftrightarrow \psi) = 1 iff V(\phi) = V(\psi)$.

4 Predicate Logic

Introduce constants and variables representing invididuals and predicates to capture the main structural building blocks of sentences. Introduce quantifiers to allow for quantified statements.

Definition

constant symbols: a, b, c, ...
variable symbols: x, y, z, ...
n-ary/n-place predicate symbols: A, B, C, reflect relations between n elements (n>0)
connectives: $\neg, \wedge, \vee, \rightarrow, ...$
quantifiers: $\forall, \exists$
round brackets (), equal sign =

Syntax: Recursive Definition

(i) If A is an n-ary predicate letter in the vocabulary of L, and each of t1,..., tn is a constant or a variable in the vocabulary of L, then At1,..., tn is a formula in L.
(ii) If ϕ is a formula in L, then $\neg\phi$ is too.
(iii) If ϕ and ψ are formulas in L, then $(\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$ are too.
(iv) If ϕ is a formula in L and x is a variable, then $\forall x\phi$ and $\exists x\phi$ is too. (v) Only that which can be generated by the clauses (i)-(iv) in a finite number of steps is a formula in L.
invalid: $a, A, \forall(Axy)$

Construction Trees

$\forall_x \forall_y ((A_{xy} \wedge B_y) \rightarrow \exists_x A_{xb})\ (iv.\forall)$
$\forall_y ((A_{xy} \wedge B_y) \rightarrow \exists_x A_{xb})\ (iv.\forall)$
$(A_{xy} \wedge B_y) \rightarrow \exists_x A_{xb}\ (iii. \rightarrow)$
$A_{xy} \wedge B_y\ (iii.\wedge)$ $\exists_x A_{xb}\ (iv.\exists)$
A_{xy} (i) B_y (i) A_{xb} (i)

Semantics

domain (D): set of entities
interpretation functions I = $\{< m, e >, < s, e >, < v, e >\}$, I(m) = e, I(s) = e, I(v) = e.
model M: consists of a domain D and an interpretation function I which conforms to: (i) if c is a constant in L, then I(c) ∈ D (ii) if B is an n-ary prpedicate letter in L, then I(B) ⊂ D
valuation function V_M:
If Aa1,...,an is an atomic sentence in L, then V_M(Aa1,...,an) = 1 iff <I(a1),...,I(an)> ∈ I(A).
...
$V_M(\forall x\phi)$ = 1 iff $V_M([c/x]\phi)$ = 1 for all constants c in L.
$V_M(\exists x\phi)$ = 1 iff $V_M([c/x]\phi)$ = 1 for at least one constant c in L.
If $V_M(\phi)$ = 1, then ϕ is said to be true in model M.

Formula vs. Sentence

A sentence is a formula in L which lacks free variables.
Sentence: Aa, $\forall x(Fx)$, $\forall x(Ax \rightarrow \exists y By)$
Not a sentence (but Formula): Ax, Fx, $Ax \rightarrow \exists y By$

5 Second-Order Logic

e.g. CR (CX: X is a predicate with the property of being a color; Rx: x is red)
$\exists X(CX \wedge Xm)$: Mars has a color.
$\exists X(Xj \wedge Xp)$: John has at least one thing in common with Peter.
$\forall x(\exists X((AX \wedge Xx) \wedge Jx) \rightarrow \exists Y(Yx \wedge CY))$: All animals that live in the jungle have a color.

Vocabulary extention

1st-order predicate variables: X, Y, Z
2nd-order predicate constants: A, B, C
e.g. AX: X is a property with the property of being an animal

Syntax: Recursive Definition

(i) If A is an n-ary first-order predicate letter in the vocabulary of L, and each of t1,..., tn are individual terms in L, then At1,..., tn is an (atomic) formula in L.
(ii) If X is a [first-order] predicate variable and t is an individual term (both constants and variables) in L, then Xt is an atomic formula in L;
(iii) If A is an n-ary second-order predicate letter/constant in L, and T1,...,Tn are first-order **unary** predicate constants, or predicate variables, in L, then AT1,...,Tn is an (atomic) formula in L;
(iv) If ϕ is a formula in L, then $\neg\phi$ is too.
(v) If ϕ and ψ are formulas in L, then $(\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$ are too.
(vii) If ϕ is a formula in L and x is a variable, then $\forall x\phi$ and $\exists x\phi$ is too.
(viii) If X is a [first-order] predicate variable, and ϕ is a formula in L, then $\forall X\phi$ and $\exists X\phi$ is too.
(viiii) Only that which can be generated by the clauses (i)-(vii) in a finite number of steps is a formula in L.
invalid: $x, X, Xab, \forall(Xa)$

Semantics

just as a 1st-order predicate denotes a set of entities, a 2nd-order predicate denotes a set of a set of entities.

6 Type theory

Tools to get to grips with frequent compositional structures in natural language (adj-n, adv-v, art-n, prep-np... combis), a higher-order logic

Definition

(i) $e, t \in T$
(ii) if $a, b \in T$, then $< a, b > \in T$
(iii) nothing is an element of T except on the basis of clauses (i) and (ii).
invalid: et, <e,e,t>, <e,<e,t>

Functional Application

If α = <e,t> and β = e then $\alpha(\beta)$ = t.
If α = <t,<t,e» and β = <t,e> then $\alpha(\beta)$ is not defined.

Semantic Types

individual: e
sentences: t
1-place predicates (intransitive verb): <e,t>
2-place predicates (transitive verb): <e,<e,t»
3-place predicates (ditransitive verb): <e,<e,<e,t»>
common nouns (e.g. dog): <e,t>
NP (e.g. the dog): e
determiners (e.g. the): <e,<e,t»
adjectives: <e,t>
adjectives as predicate modifiers(e.g. *happy* dog): «e,t>,<e,t»
adverbs(predicate modifier): «e,t>,<e,t»
sentence modifier(e.g. not): <t,t>
function(entity to entity) (e.g. the farther of): <e, e>
1-place 2nd-order predicate: «e,t>,t>
2-place 2nd-order predicate: «e,t>,«e,t>,t»

Syntax: Recursive Definition

(i) If α is a variable or a constant of type a in L, then α is an expression of type a in L.
(ii) If α is an expression of type <a,b> in L, and β is an expression of type a in L, then $(\alpha(\beta))$ is an expression of type b in L.
(iii) If ϕ and ψ are formulas in L, then so are $\neg\phi, (\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$.
(iv) If ϕ is an expression *of type t* in L and v is a variable (of arbitrary type a), then $\forall v\phi$ and $\exists v\phi$ are expression of type t in L.
(v) If α and $beta$ are expressions in L which belong to the same type, then $(\alpha = \beta)$ is an expression of type t in L.
(vi) Every expression L is to be constructed by means of (i)-(v) in a finite number of steps.
formulas: expressions of type t.
Difference to Predicate Logic:
Jumbo befriends Maya.
Predicate Logic: Bjm
Type-theoretic logic: (B(m))(j) or alternatively B(m)(j)

Semantics

Truth valuations via pariticular interpretation functions defined for different types of expressions. e.g. interpretation function I for which it holds that: I(W)(d) = 1 iff d∈W, otherwise 0.

7 Lambda Calculus

To represent parts of sentences or predicates in a fully compositional account. Allows to capture the compositionality of language

Syntax

Add another clause to the type-theoretic language syntax
(vii) If α is an expression of type a in L, and v is a variable of type b, then $\lambda v(\alpha)$ is an expression of type <b, a> in L.

Lambda-Abstraction

We say that $\lambda v(\alpha)$ has been formed from α by abstraction over the formerly free variable v. Hence, the free occurrences of v in α are now bound by the λ-operator λx.
e.g. expression: S(x) of type t
λ-abstraction: λx(S(x)) of type <e,t>
λx(x) of type <e,e>
λx(B(y)(x)) of type <e,t>
λX(X(a) $\wedge$ X(b)) of type «e,t>,t>

Lambda-Conversion

remove the λ-operator and plug an expression into every occurrence of the variable which is bound by the λ-operator.
e.g. λx(S(x))(c) = S(c)
λx(λy(A(y)(x)))(c)(d)=λy(A(y)(c))(d)=A(d)(c)
λ-conversion is only valid when vairable v is not bound by a quantifier $\forall$ or $\exists$

Modelling Compositionality

John smokes: λx(S(x))(j)=S(j)
smokes: λx(S(x))
smokes and drinks: λx(S(x)$\wedge$D(x)) Jumbo is grey: λx(G(x))(j)=G(j)
is grey: λx(G(x))
Jumbo is: λX(X(j)
is: λX(λx(X(x)))

Truth Valuation

For all entities d in the domain D it holds that h(d)=1 iff I(W)(d)=1. This illustrates that the denotation if $\lambda x(W(x))$ is indeed the same as one would expect for just the word walks represented by W.