Instructions

1. Please fill in your answers on the exam sheets (5 pages).
2. Exam duration: 2.5 hours
3. You may use any pre-prepared material.
4. Please write your student number here: ________________.
5. The students who have not yet filled in the evaluation form on the web are kindly requested to fill in the enclosed printed version after completing the exam.

Good luck!

Question 1  (5+5+4+8=22 points + 5 bonus points)
Consider the following sentences, with the assumed constituent structures:

(1.0) The room is [clean].
(1.1) The room is [not clean].
(1.2) The room is [almost clean].
(1.3) The room is [completely clean].
(1.4) The room is [[not completely] clean].
(1.5) The room is [[almost completely] clean].
(1.6) The room is [not [completely clean]].
(1.7) The room is [almost [completely clean]].

a. Write down the types for the following occurrences of words from these sentences:
   - The word not in (1.1): __________
   - The word almost in (1.2): __________
   - The word completely in (1.3)-(1.7): __________
   - The word not in (1.4): __________
   - The word almost in (1.5): __________
   - The word not in (1.6): __________
   - The word almost in (1.7): __________

b. Write down all entailments between the sentences in (1.0)-(1.5):

________________________________________________________________________

c. In order to show a truth-conditional distinction between (1.4) and (1.6), show an entailment with one of the other sentences in (1.0)-(1.7) that one of the sentences (1.4) and (1.6) supports and the other does not:
   _____ ⇒ _____; _____ ⊬ _____.
d. (Bonus:) Can you find a similar truth-conditional distinction between (1.5) and (1.7)? If you can show an entailment that one of the sentences (1.5) and (1.7) supports and the other does not:

\[ \text{truth-value for (1.5)} \Rightarrow \text{truth-value for (1.7)} \]

\[ \text{truth-value for (1.5)} \not\Rightarrow \text{truth-value for (1.7)} \]

e. What assumptions should we adopt about the denotations of *almost* in sentences (1.2) and (1.5) that would account for the entailments you described in your answer to b? Complete the following statements:

- *almost* in (1.2) must denote a function \( f \) that for each argument \( A \) of type _____ satisfies: _______________________________

- *almost* in (1.5) must denote a function \( g \) that for each argument \( B \) of type _____ satisfies: _______________________________

**Question 2** (6+6+6+6+3+3=30 points + 5 bonus points)

Consider the following sentences, with the assumed constituent structures:

(2.1) \([\text{Exactly one]} \text{ [singer [who dances]]} \text{ retires.}\)
(2.2) \([\text{John, [who dances], ] retires.}\)

We assume the following types for the lexical expressions in (2.1)-(2.2):\(^1\)

<table>
<thead>
<tr>
<th>Lexical Expression</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>exactly one</td>
<td>((et)(et)t)</td>
</tr>
<tr>
<td>singer</td>
<td>(et)</td>
</tr>
<tr>
<td>dances</td>
<td>(et)</td>
</tr>
<tr>
<td>retires</td>
<td>(et)</td>
</tr>
<tr>
<td>John</td>
<td>(e)</td>
</tr>
</tbody>
</table>

a. Assume that the denotation of *John* is the entity \( j \), and that the sets characterized by the denotations for *singer*, *dances* and *retires* are \( S \), \( D \) and \( R \) respectively. Write down appropriate truth-values for (2.1) and (2.2) using these denotations:

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Truth-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2.1)</td>
<td></td>
</tr>
<tr>
<td>(2.2)</td>
<td></td>
</tr>
</tbody>
</table>

b. Give two different appropriate types for the word *who* in (2.1) and (2.2):

<table>
<thead>
<tr>
<th>Word</th>
<th>Type in (2.1)</th>
<th>Type in (2.2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>who</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b. Give two different appropriate types for the word *who* in (2.1) and (2.2):

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Type for who</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2.1)</td>
<td></td>
</tr>
<tr>
<td>(2.2)</td>
<td></td>
</tr>
</tbody>
</table>

c. Using these types and the denotations you gave for (2.1) and (2.2), write the two appropriate \( \lambda \)-terms for the meaning of *who* in the two sentences:

<table>
<thead>
<tr>
<th>Sentence</th>
<th>( \lambda )-Term for who</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2.1)</td>
<td></td>
</tr>
<tr>
<td>(2.2)</td>
<td></td>
</tr>
</tbody>
</table>

Consider now the following sentence, with the assumed constituent structure:

(2.3) \([[[\text{Exactly one]} \text{ [singer], [who dances]]}, \text{ retires.}\)

\(^1\text{exactly one is assumed to be a lexical expression.}\)
d. We assume that (2.3), with the comma intonations after singer and dances, is not equivalent to (2.1), without these intonations. Describe a model that supports this intuition by giving a definition to the sets $S$, $D$ and $R$:

$$S = \text{______________________}$$

$$D = \text{______________________}$$

$$R = \text{______________________}$$

e. Write down now a general truth-value for (2.3) using the sets $S$, $D$, and $R$ in any model (not necessarily the one you described in 2d):

truth-value for (2.3): ____________________________

f. Propose now a type for the word who in (2.3):

type for who in (2.3): ____________________________

g. (bonus question:) Suppose that you would be required to find an appropriate denotation for the word who in (2.3). What problem would you encounter?

---

**Question 3**  (6+4+5+5+10=30 points)

Consider the following sentences, with the assumed constituent structures:

(3.0) [The picture] disappeared.
(3.1) [The [picture [of John]]] disappeared.
(3.2) [[John 's] picture] disappeared.
(3.3) [A picture] disappeared.

We assume that the definite article the is of type $(et)e$, and its denotation the is defined as follows, for every one-place predicate $P_{et}$ characterizing a set $A \subseteq D_e$:

$$\text{the}(P) = \begin{cases} 
    a & A = \{a\} \\
    \text{undefined} & \text{otherwise}
\end{cases}$$

Thus – under this treatment the truth-value of (3.0) is:

true – if there is a unique picture and that picture disappeared;
false – if there is a unique picture and that picture did not disappear;
undefined – if there is no unique picture.

Let us now assume that the sentences (3.1) and (3.2) are equivalent. Thus, according to the truth-conditionality criterion, in all intended models the truth-values of these sentences should come out the same (true, false or undefined). Our task in this question is to satisfy this requirement.

a. We assume that the types of the words picture, John and disappeared are $et$, $e$ and $et$ respectively. Complete the proper types for the words $of$ and 's in (3.1) and (3.2):

type for $of$ in (3.1): ____________________________

type for 's in (3.2): ____________________________

b. Let $of$ be the denotation of the word $of$ in (3.1), of the type you suggested in 3a. Note the entailment (3.1)$\Rightarrow$(3.3). What semantic restriction(s) should the denotation $of$ satisfy in every model, in order to account for this entailment? Write it formally.
c. Note the equivalence \((3.1) \Leftrightarrow (3.2)\). Make sure now that this equivalence is captured by defining the denotation of the word ‘s. Write down this denotation as a \(\lambda\)-term of the type you suggested in 3a, using the constants \(\text{of}\) and \(\text{the}\) of the types defined above:

\(\lambda\)-term for ‘s in (3.1): 

In some cases there seems to be a strong relation between the meaning of the words \(\text{of}\) and ‘s and the meaning of the verb \(\text{own}\). For instance, consider the tautological status of the following sentences:

(3.4) John [owns [[John ‘s] dog]].
(3.5) John [owns [the [dog [of John]]]].

d. We standardly assume that the type of the transitive verb \(\text{own}\) is \(e(tet)\) and that it denotes a function \(\text{own}\). Taking the tautologies (3.4) and (3.5) into account, write a \(\lambda\)-term for the denotation \(\text{of}\) of the word \(\text{of}\), in terms of the constant \(\text{own}\) of type \(e(tet)\).

\(\lambda\)-term for \(\text{of}\) in (3.2): 

e. We now want to make sure that the truth-value of (3.4) is \text{true} or \text{undefined} in all intended models. We will do that by using the denotations you gave above for ‘s, in terms of a constants \(\text{own}\) and \(\text{the}\). First, write down the truth-value for (3.4) in terms of the constants \(j\), \(\text{own}\), ‘s and \(\text{dog}\):

Now, in this formula, replace \(s’\) by the \(\lambda\)-term you gave in 3c using the constants \(\text{of}\) and \(\text{the}\).

Do the necessary reductions in order to get a normal form:

Replace \(\text{of}\) by the definition you gave in 3d using the constant \(\text{own}\):

Do the necessary reductions in order to get a normal form:

Explain informally why the formula you now got is \text{true} or \text{undefined} in all intended models that respect the definition we gave for \(\text{the}\).

**Question 4** (9+9=18 points)

Consider the Haskell lexicon on the next page. In this question you will be asked to add entries to this lexicon with appropriate categories and denotations, in order to allow parsing and truth-value assignment for the following sentences.

(4.1) Luke is neither evil nor alien nor hairy.
(4.2) Neither exactly one nor more than two jedi are hairy or alien.
(4.3) Han neither killed Luke nor ran.
(4.4) Between one and four aliens are hairy.

a. Define entries for "Neither" and "nor" such that (4.1), (4.2) and (4.3) can be parsed:

, entry "neither" 
, entry "nor" 

|
b. The lexicon contains an entry for \textit{and} that takes two numbers and returns a pair of numbers. Its category takes a \texttt{num} left and a \texttt{num} right and returns a pair \texttt{num:*num}. Its function takes a \texttt{Int} and another \texttt{Int} and returns a pair \texttt{(Int,Int)}. Use such a pair of numbers for defining an \textit{inclusive} interval, in order to give the denotation for "between". Add an entry for \textit{between} that makes it possible to parse (4.4).

\begin{verbatim}
, entry "between" __________ between
between :: __________
\end{verbatim}

\texttt{Lexicon.hs}

\begin{verbatim}
-- takes a function that characterizes a set and returns that set
toList :: (E->T) -> [E]
toList f = filter f entities
-- takes a set and returns its characteristic function
charf :: [E] -> E -> T
charf list x = x 'elem' list
-- takes a function that characterizes a set and returns its cardinality
card :: (E->T) -> Int
card f = length (toList f)
-- category for determiners
det = ((s://(np:\s)):/n)

lexicon :: Lexicon

lexicon =
{ entry "Luke" np ( Luke :: E )
, entry "Han" np ( Han :: E )
, entry "Leia" np ( Leia :: E )
, entry "evil" n ( (charf evil_set) :: E->T )
, entry "hairy" n ( (charf hairy_set) :: E->T )
, entry "alien" n ( (charf alien_set) :: E->T )
, entry "aliens" n ( (charf alien_set) :: E->T )
, entry "jedi" n ( (charf jedi_set) :: E->T )
, entry "ran" (np:\s) ( (charf ran_set) :: E->T )
, entry "killed" (np:\s:/np) ( (charf2 killed_relation) :: E->E->T )
, entry "kissed" (np:\s:/np) ( (charf2 kissed_relation) :: E->E->T )
, entry "ls" (np:\s:/n) ( (\x -> x) :: (E->T)->E->T )
, entry "are" (np:\s:/n) ( (\x -> x) :: (E->T)->E->T )
, entry "than" (n:\n:/n) ( (\x -> x) :: (E->T)->E->T )
, entry "or" (n:\n:/n) ( (\x -> x) :: (E->T)->E->T )
, entry "and" (n:\n:/n) ( (\x -> x) :: (E->T)->E->T )
, entry "not" (n:/n) ( compl1 :: (E->T)->E->T )
, entry "one" num (1 :: Int)
, entry "two" num (2 :: Int)
, entry "three" num (3 :: Int)
, entry "some" det (pred_det (>0))
, entry "exactly" (det :/ num) (\x -> pred_det (=x) )
, entry "more_than" (det :/ num) (\x -> pred_det (>x) )
, entry "less_than" (det :/ num) (\x -> pred_det (<x) )
, entry "more" (det :/ n) "more"
, entry "and" (num:\((num:*num):/num)) num_and
}

pred_det :: (Int->T) -> (E->T) -> (E->T) -> T
pred_det p f g = p ( card (f \ g) )

num_and :: Int -> Int -> (Int,Int)
num_and a b = (a,b)

more :: (E->T) -> (E->T) -> (E->T) -> T
more a b c = card (a \ c) - card (b \ c) > 0
\end{verbatim}