Fragment: qanda.pl

Questions and Answers
Abstract

The fragment `film_db.pl` uses the type-logical machinery to develop a natural language database query system. In this session, we discuss the underlying grammatical architecture. In section 1, we focus on the syntax of yes-no questions and constituent questions. In section 2, we go into the meaning of these two types of interrogative sentences, taking the relation between questions and their answers as our starting point.

If you would like to find out more about this approach, you can have a look at the following paper, which we draw on for section 2 and which discusses the relevant issues in greater depth:

RAFFAELLA BERNARDI and RICHARD MOOT (2000), A natural language interface to a query system. 2nd Workshop on Inference in Computational Semantics (ICoS-2), Schloss Dagstuhl. [Download Postscript]

Michael Moortgat, UCLA Minicourse, November 2000
## Contents

1 Syntax ................................................................. 4  
   1.1 Yes-no questions ........................................... 5  
   1.2 Constituent questions ...................................... 11  
2 Semantics ............................................................ 16  
   2.1 Yes-no questions ........................................... 17  
   2.2 Constituent questions ...................................... 18  
Solutions to the exercises ........................................ 22
1. Syntax

In this fragment, we want to distinguish the following types of sentences:

\[ s : \text{Simple declarative sentences. Examples:} \]

Alice teases Tweedledum.
Nobody stole the tarts.
1. Syntax

In this fragment, we want to distinguish the following types of sentences:

$s$: Simple declarative sentences. Examples:

Alice teases Tweedledum.
Nobody stole the tarts.

$q$: Yes-no questions, also known as *polar interrogatives*. Examples:

Does Alice tease Tweedledum?
Did anybody steal the tarts?
1. Syntax

In this fragment, we want to distinguish the following types of sentences:

\[ s \]: Simple declarative sentences. Examples:

Alice teases Tweedledum.
Nobody stole the tarts.

\[ q \]: Yes-no questions, also known as polar interrogatives. Examples:

Does Alice tease Tweedledum?
Did anybody steal the tarts?

\[ wh \]: Constituent questions. Examples:

Who does Alice tease?
Who stole the tarts?
1.1. Yes-no questions

Polar interrogatives, in English, are formed with the auxiliary verb do:

- The auxiliary appears in initial position, and carries the tense information — present or past: we ignore this distinction here;
- The main verb appears in the infinitive form.

We introduce the basic type $inf$ for infinitives, and interpret these as properties, $\text{Ind} \rightarrow \text{Bool}$. For the time being, we also introduce a basic type $q$ for yes-no questions. In section 2, we come up with an appropriate denotation type for $q$.

Types for verbs In the lexicon, next to the tensed main verbs, we put the infinitival forms, obtained by replacing the subtype $np\backslash s$ by $inf$.

irritates : $(np\backslash s)\backslash np$ → irritate
irritate : $inf\backslash np$ → irritate
puts : $((np\backslash s)\backslash pp)\backslash np$ → put
put : $(inf\backslash pp)\backslash np$ → put
Typing for auxiliary ‘do’ What would be an appropriate type for the auxiliary verb? We want the auxiliary to combine with a subject and an infinitival phrase (in that order) to form a yes-no question \( q \). Here is a first attempt.

\[
\text{does} : q/(np \bullet inf) - \lambda x. (?((\pi_1 x) (\pi_0 x)))
\]

In this solution, the subject and the infinitive are taken together as a product type \( np \bullet inf \). Semantically, this is a pair of an \( np \) type meaning \((\text{Ind})\) and an \( inf \) type meaning \((\text{Ind} \rightarrow \text{Bool})\).

In the body of the lexical program for the auxiliary, we apply the second projection of that pair (i.e. the \( inf \) property) to the first projection (i.e. the term for the subject \( np \)). The resulting proposition is given to a question operator ‘?’.

For the time being, we give no details as to what this operator does — we cannot, since we haven’t decided what kind of semantic domain would be appropriate for \( q \).
Sample derivation A natural deduction derivation, and the meaning assembly (in Fitch style).

1. \( \text{does} : q/np \bullet inf \rightarrow \lambda z_1.\left(\pi_1 z_1 \ (\pi_0 z_1)\right) \)
2. \( \text{alice} : np \rightarrow \text{alice} \)
3. \( \text{irritate} : inf/np \rightarrow \text{irritate} \)
4. \( \text{tweedledum} : np \rightarrow \text{tdum} \)
5. \( \text{irritate} \circ \text{tweedledum} : inf \rightarrow (\text{irritate tdum}) \)
6. \( \text{alice} \circ (\text{irritate} \circ \text{tweedledum}) : np \bullet inf \rightarrow \langle \text{alice}, \langle \text{irritate tdum} \rangle \rangle \)
7. \( \text{does} \circ (\text{alice} \circ (\text{irritate} \circ \text{tweedledum})) : q \rightarrow (\ ? ((\text{irritate tdum}) \text{ alice}) \)
Exercise 1. The above type assignment for the auxiliary \texttt{does} takes together the subject and the infinitive as a product type $np \bullet inf$. Out of the subtypes $q$, $np$ and $inf$, you could also construct a purely implicational type formula (with the implication ‘/’) that would derive the same yes-no questions.

Make a lexical entry with this implicational type formula for the auxiliary \texttt{did}. You will have to adapt the lexical meaning program that goes with it! Check your answer with a derivation for the sample sentence on the previous page: the meaning you obtain (after simplification) should be the same for these two alternative typings.
Lifting the typing for the auxiliary  The two typings we have been considering have a
common problem: they only allow proper names (like ‘Alice’) or definite descriptions
(like ‘the Red Queen’) in the subject position. But of course, we also want to have
full generalized quantifier expressions in that position — compare:

Does Alice irritate Tweedledum?
Does everyone irritate Alice?

Here is an improved type assignment for our product-based typing. In the lexical
meaning program, it is now the GQ meaning that is applied to the $\text{inf}$ property.

\[
\text{does} : q/((s/(np\,s)) \bullet \text{inf}) - \lambda x. (? ((\pi_0 \, x) \, (\pi_1 \, x)))
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \\
\text{alice}
\end{array}
\vdash \text{inf} \quad [/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash \text{inf} \
[/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash (s/(np\,s)) \bullet \text{inf} \
[/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash q
[/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash (s/(np\,s)) \bullet \text{inf} \
[/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash q
[/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash (s/(np\,s)) \bullet \text{inf} \
[/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash q
[/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash (s/(np\,s)) \bullet \text{inf} \
[/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash q
[/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash (s/(np\,s)) \bullet \text{inf} \
[/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash q
[/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash (s/(np\,s)) \bullet \text{inf} \
[/E]
\]

\[
\begin{array}{c}
\text{everyone} \\
\inf/np \\
\text{irritate} \circ \text{alice}
\end{array}
\vdash q
[/E]
\]
Exercise 2. Compute the meaning assembly for the following derivation. Simplify where possible.

1. \( \text{does} : q/(s/(np\backslash s)) \bullet in\!f \)

2. \( \text{alice} : np \)

3. \( r_0 : np\backslash s \)

4. \( \text{alice} \circ r_0 : s \)

5. \( \text{alice} : s/(np\backslash s) \)

6. \( \text{irritate} : in\!f / np \)

7. \( \text{tweedledum} : np \)

8. \( \text{irritate} \circ \text{tweedledum} : in\!f \)

9. \( \text{alice} \circ (\text{irritate} \circ \text{tweedledum}) : (s/(np\backslash s)) \bullet in\!f \bullet I (5, 8) \)

10. \( \text{does} \circ (\text{alice} \circ (\text{irritate} \circ \text{tweedledum})) : q \)

Exercise 3. Perform the change from subtype \( np \) to \( s/(np\backslash s) \) also in the purely implicational type assignment you have found for \text{did} in a previous exercise. Adapt the lexical meaning program accordingly, and give the derivation and meaning assembly of the question above with your implicational type for \text{does}.

1.2. Constituent questions

We are ready now to address constituent questions — also known as *wh*-questions, because they are introduced by *wh* words (like *who*, *what*, *why*, *how*, ...). The distinction we have to make is between constituent questions regarding the subject (as in a), versus non-subject constituent questions (as in b and c).

a. Who irritates the Red Queen?
b. Who does the Red Queen irritate?
c. What did Alice put on the table?

Non-subject questions As with the simple yes-no questions of the previous section, in the non-subject cases we find inversion of the subject and the auxiliary. As a matter of fact, with an *np* hypothesis, the material following the *wh* words would be a constituent of type *q*:

\[
\begin{align*}
\text{does the Red Queen irritate} & \quad np \vdash q \\
\text{did Alice put} & \quad np \text{ on the table} \vdash q
\end{align*}
\]
This is valuable information for the typing of the *wh* words *who* and *what*! We want these words to produce a phrase of type *wh* with a *q* phrase that is missing a *np* hypothesis somewhere. We can characterize the structural positions accessible for the *np* hypothesis in terms of the structural package $P1/P2$: structural inferences that allow extraction from right branches, as we have seen in our discussion of relative clauses.

$$
(A \bullet B) \bullet \Diamond C \vdash A \bullet (B \bullet \Diamond C) \quad [P1]
$$

$$
(A \bullet C) \bullet \Diamond B \vdash (A \bullet \Diamond B) \bullet C \quad [P2]
$$

Here are type assignments with the required properties. As with the yes-no questions, we associate the *wh* words with a semantic operation *WH*, but we refrain from going into detail as to what this operation does — that issue will be taken up in section 2.

- **what**: $wh/(q/\Diamond \Box np) - WH$
- **who**: $wh/(q/\Diamond \Box np) - WH$
Exercise 4. Compute the meaning assembly for the derivation below. You can ignore the derivational semantics for the unary operators: use $np$-type semantic variables for the hypotheses $p_1$ and $r_0$, and associate the $\Box$ Elimination step with the identity operation, and the $\Diamond$ Elimination step with substitution.

$$
\frac{
\text{alice} \quad [r_1 \vdash np \backslash s]^3 \\
vnp \quad [\Box E] \\
\text{irratiate} \quad [p_1 \vdash np]^2 \\
\text{inf} \quad [\Box E]
}{
q/\left(s/(np\backslash s)\right) \cdot \text{inf}
}

$$

$$
\frac{
\text{does} \\
\text{alice} \quad \langle p_1 \rangle \vdash s/(np\backslash s) \\
\text{alice} \quad \langle p_1 \rangle \vdash \text{inf} \\
\text{irratiate} \quad \langle p_1 \rangle \vdash \text{inf} \\
\text{inf} \quad [\Box E]
}{
\text{q/\left(s/(np\backslash s)\right) \cdot \text{inf}}
}

$$

$$
\frac{
\text{does} \circ \langle p_1 \rangle \vdash q \\
\text{does} \circ \langle p_1 \rangle \vdash q \\
\text{[P1]} \\
\text{[P1]} \\
\text{[\Diamond E]^2}
}{
\text{does} \circ \langle p_1 \rangle \vdash q
}

$$

$$
\frac{
\text{does} \circ \langle p_1 \rangle \vdash q \\
\text{[I]^1}
}{
\text{[E]}
}

$$

$$
\frac{
\text{who} \quad [r_0 \vdash \Box np]^1 \\
\text{wh} \quad [q/\Box np]
}{
\text{who} \circ (\text{does} \circ \langle p_1 \rangle) \vdash q
}

$$

$$
\frac{
\text{who} \circ (\text{does} \circ \langle p_1 \rangle) \vdash q \\
\text{[E]}
}{
\text{wh} \\
\text{q/\Box np}
}

$$
**Left versus right branches** In previous exercises, we have compared two alternative typings for the auxiliary: one (your entry for *does*) where the auxiliary takes as an argument a product formula combining the subject and the infinitive phrase, and one (your entry for *did*) where subject and infinitive are consumed in that order as arguments of ‘/’ implications.

From a semantic point of view, there is nothing to choose between these two alternatives. But structurally, they make different claims about the configuration of the *q* phrase:

- with the product-based type, the *q* phrase as a whole is right-branching, and the subject is situated on a **left** branch;

- with the implicational type, the *q* phrase as a whole is right-branching, and the subject is situated on a **right** branch.
In combination with the extraction package $P1/P2$, the alternative typings for the auxiliary make different predictions. If did gets the implicational type, the subject would be accessible for extraction, and the ungrammatical $a$ question below would be derivable (you can try it). What we need instead, if we want to question the subject, is $b$.

\begin{enumerate}
\item \textit{*Who did irritate Tweedledum?}
\item Who irritates Tweedledum?
\end{enumerate}

**Exercise: subject questions** As the $b$ example above shows, subject questions do not involve the auxiliary/infinitive combination: the \textit{wh} word in this case directly combines with a tensed verb phrase (type $np\backslash s$), and there is no need for structural reasoning. Provide the required type assignment for the $b$ example.
2. Semantics

In section 1, we have treated $q$ and $wh$ as atomic types, and we have refrained from specifying what semantic domains would be appropriate for the interpretation of these types. It is time to address these issues now. We will follow the strategy of the ‘structured meaning’ approach to questions:

> Question meanings are functions that, when applied to the meaning of the answer, yield a proposition.

You can read more about this approach in Manfred Krifka’s 1999 paper *For a structured account of questions and answers*, which is available on-line.

In subsection 2.1, we look at polar interrogatives, with their answers ‘Yes’ or ‘No’. Then, in subsection 2.2, we turn to *wh* questions, with constituent answers.
2.1. Yes-no questions

Polar interrogatives, typically, have answers ‘Yes’ or ‘No’. We can consider these answers as propositional operators with denotations in \( \text{Bool} \rightarrow \text{Bool} \). A negative answer reverses the truth value; a positive answer preserves it. The type assignment below is compatible with this semantic characterization.

\[
\begin{align*}
\text{no} & : s \downarrow s \rightarrow \lambda p. \neg p \\
\text{yes} & : s \downarrow s \rightarrow \lambda p. p
\end{align*}
\]

Given this type assignment for the answers to polar interrogatives, we can now treat \( q \) as a macro for \( s/(s\downarrow s) \), with a denotation in \( (\text{Bool} \rightarrow \text{Bool}) \rightarrow \text{Bool} \). Below you find the adjusted lexical entry for the auxiliary. The parameter \( y_0 \) here is for the answer (with denotation in \( \text{Bool} \rightarrow \text{Bool} \)).

\[
\text{does} : (s/(s\downarrow s))/((s/(np\downarrow s)) \bullet inf) \rightarrow \lambda x_0. \lambda y_0. (y_0 ((\pi_0 x_0) (\pi_1 x_0)))
\]
Exercise 1. Compute the meaning assembly for the following question-answer combination.

Q. Does Alice irritate Tweedledum?
A. No.

\[
\frac{\text{does}}{(s/(s/s))/(s/(np/s)) \bullet \text{inf}} \\
\frac{\text{alice}}{np} \quad \frac{[p_1 \vdash np \backslash s]^2}{[E]} \\
\frac{\text{alice} \circ p_1 \vdash s}{[I]^2} \\
\frac{\text{irritate}}{inf/np} \quad \frac{\text{tweedledum}}{np} \quad \frac{\text{irritate} \circ \text{tweedledum}}{\vdash \text{inf}} \\
\frac{\text{alice} \circ (\text{irritate} \circ \text{tweedledum}) \vdash (s/(np/s)) \bullet \text{inf}}{[I]} \\
\frac{\text{does} \circ (\text{alice} \circ (\text{irritate} \circ \text{tweedledum})) \vdash s/(s/s)}{[E]} \\
\frac{(\text{does} \circ (\text{alice} \circ (\text{irritate} \circ \text{tweedledum}))) \circ \text{no} \vdash s}{[E]}
\]

2.2. Constituent questions

The answers to \textit{wh} questions are constituents: noun phrases in the case of the questions pronouns \textit{who}, \textit{what} — proper names as in A1, definite descriptions as in
A2, or generalized quantifiers as in A3.

\[ Q. \quad \text{Who does Alice irritate?} \]
\[ A1. \quad \text{Tweedledum.} \]
\[ A2. \quad \text{The Red Queen.} \]
\[ A3. \quad \text{Nobody.} \]

Given this relation between \( wh \) questions and their answers, the appropriate semantic domain for such constituent questions would be

\[
((\text{Ind} \rightarrow \text{Bool}) \rightarrow \text{Bool}) \rightarrow \text{Bool}
\]

that is, functions from generalized quantifier type meanings (sets of properties) to truth values. Reading \( wh \) as a macro abbreviation for \( s/((s/np)\backslash s) \), we have a syntactic type that indeed takes its denotation in this semantic domain.

For the direct object questions above, unfolding the macro definitions for \( q \) and \( wh \) yields the following adjusted type assignment.

\[
\text{who} : \quad wh/(q/\Diamond \Box np))
\]
wh/((s/(s\ s))/◊□np)  (macro for q)
(s/((s/\ np)\ s))/((s/(s\ s))/◊□np)  (macro for wh)

We can have enough information now to spell out the meaning program for the operation WH which we associated with the *wh* pronoun.

\[
\text{WH} = \lambda x_0.\lambda y_0.\text{BODY}
\]

To specify the BODY of this program, we have to keep in mind that the parameter \(x_0\) denotes a function from \(np\) type meanings to \(q\) type meanings, i.e.

\[
\text{Ind} \rightarrow ((\text{Bool} \rightarrow \text{Bool}) \rightarrow \text{Bool})
\]

and that \(y_0\) (the parameter for the answer) has a generalized quantifier type denotation. The component BODY as a whole should have a propositional denotation. We can put together the pieces of this puzzle as follows (with \(z_0\) in Ind and \(x_1\) in Bool):

\[
\text{WH} = \lambda x_0.\lambda y_0.\lambda z_0.((x_0 z_0) \lambda x_1. x_1)
\]
Exercise 2. Try out this program on the dialogue:

Q. Who does Alice irritate?

Exercise 3. The type assignment above covers the non-subject cases of *wh* questions. Unfold the macro for the *wh* type in your earlier solution for subject *wh* questions, and adjust the meaning recipe accordingly. Test your new lexical entry with a derivation for the following dialogue:

Q. Who irritates Alice?
A. Everybody.
Solutions to the exercises

Exercise 1. Here is our entry. In the meaning program, the parameter $x_0$ is for the subject, $y_0$ for the infinitive.

\[ \text{did} : (q/inf)/np - \lambda x_0.\lambda y_0. (\ ? \ (y_0 \ x_0)) \]

Click on the example to fetch a derivation:

\[ \text{Did Alice irritate Tweedledum}? \vdash q \]
Exercise 2. Click on the example for a semantically annotated derivation.

Does Alice irritate Tweedledum? \( \vdash q \)
Exercise 3. The higher-order type for \textit{did}:

\[
\text{did} : (q/\text{inf})/(s/(np\backslash s)) \rightarrow \lambda x_0.\lambda y_0.(? (x_0 \, y_0))
\]

Click on the example for a semantically annotated derivation.

Did Alice irritate Tweedledum? \vdash q
Exercise 4. Click on the example for a semantically annotated derivation.

Who does Alice irritate? ⊢ wh
Exercise 1. Your solution for this exercise should produce the following result, after simplification. Click on the sentence to fetch the derivation.

\[ \neg((\text{irritate } tdum) \ alice) \]

Does Alice irritate Tweedledum? No. \(\vdash s\)
Exercise 2. Here are the two question/answer pairs:

Who does Alice irritate? Tweedledee. ⊢ s

Who does Alice irritate? Nobody. ⊢ s
Exercise 3.

Who irritates Alice? Everybody. ⊢ s