Combined queries

As shown above, you can query for individual tokens. But what if you need more detail, maybe from different layers? Let us consider the following examples:

1. The Italian token *io* used by only male informants
2. Different spelling forms of the German token *was* used as a relative pronoun
3. Occurences of the Italian verb form *sono* preceded by a personal pronoun

Before looking at how to build those queries, let us describe their structure:

1. The first example describes a specific value for a token that is embedded in a message that fulfills specific criteria. We thus have two conditions on different layers (token and message).
2. The second example describes a series of spelling variants as they are found on the token level. The same token has to have a specific value as a normalisation and a specific part of speech. We thus look only at tokens but with two different attributes (annotations for lemma and PoS).
3. In the third example we look for a specific token *sono* and for a PoS annotation (personal pronoun) that are in a specific relationship to one another, i.e. one is following the other directly.

The queries for these examples are the following:

1. To find *io* written by males, we query for: `tok="io" & meta::sex="M"`. That reads as: a token with the contents *io* and the gender *m*.
2. In the second example we are looking for different spelling variants for the standard spelling *was*. We can find these forms as a relative pronoun by using the normalized layers and the PoS annotation. In the normalized layer, we are looking for the spelling form *was* regardless of the original respresentation. In the PoS annotation we look for a relative pronoun. This results in the query: `tok="was" & pos="PRELS" & #1 _=_ #2`. We can translate that as: the token has to be *was* and the PoS-Annotation has to be *PRELS* and the two annotations have to be found on the same token *#1 _=_ #2*. Please keep in mind that this is the syntax for subcorpora tagged with TreeTagger. The RFTagger uses a more precise annotation for relative pronouns, e.g. PRO.Rel.Subst.Nom.Sg.Neut. The query would thus look like: `tok="was" & pos=/PRO.Rel.*/ & #1 _=_ #2`.
3. In the third example we look for two tokens, one directly following the other. Here, we could use one of the normalisations, too, i.e. `gloss` or we could use the token. This choice depends on what we want to find. If we want to after the spelling *est-ce que* used by the informant, we query for `tok=/…/`. If, on the other hand, we want to include unconventional spellings like *sno*, we have to use `gloss=/…/`. Let us use the first option, which gives us the following query: `pos="PRO:pers" & tok="sono" & #1 . #2`, which we can read as: a first token has to be a personal pronound and a second one has to be *sono*. The expression `#1 . #2` means the first token has to directly precede the second one.

That much for the examples. But how can you remember all of these options? You do not have to, since ANNIS offers you lots of support in creation the queries.