Minutes meeting 2016-05-02

Agenda

- Contributor CLA
- Discuss project plan
  - Discuss sprint duration and demo meetings
- Discuss research report

Contributor CLA

Usually done by means of an email, which we will receive one of these days from Hendrik.

Discuss project plan

Hendrik has not yet read the new version. He will email his feedback later. This means that we still need to discuss sprint duration, demo meetings and whether Hendrik formally signs of the project plan.

Discuss research report

- The overall structure is ok. Most comments concern writing things down in a more concise and precise manner.
- We also make too many general claims, such as "most modern programming languages ..." and "many parsing algorithms ...". Suggestion: "traditional parsing algorithms (such as LL1, LR, ...) ...".
- The report does not give the impression we know the internals of Spoofax or how languages are executed in general. To resolve this, we need to expand the Spoofax section (section 1) with aspects from the developer experience.
- For the completeness, we need to mention that Spoofax can also compile instead of interpret, or even use Stratego. We should then mention that we're opting for interpretation considering the nature of a REPL.

Feedback per section

- Introduction:
  - Name the purpose of the research report and *then* the structure, including section numbers.
  - The sentence about rapid prototyping is off; it seems to suggest a necessity that does not exist.
  - Do not mention "generic solutions" and "good implementation": this is too vague. Suggestion: "(mostly) language-agnostic".
- Spoofax section (section 1):
  - There are four major aspects on a conceptual level concerning execution of programming languages: syntax, static semantics, (optionally) transformations, interpretation/compilation).
  - Gerlof explained we know more than the report shows and explained his experiments in spoofax-test and Spoofax's services. He then drew a conceptual overview of executing a

language in Spoofax. TODO: explain schema. What was missing, however, is an interpreter or a transform service to apply interpretation by transformation. Another possibility here is DynSem.
    - This needs to be added after the conceptual piece that is also to be added in the Spoofax section.
    - "parts of language specification" versus "parts of Spoofax". Suggestion: "language specification defines the following ..."
    - The conceptual stages mentioned earlier should be discussed prior to Spoofax's parts.
    - The examples given cross several domains (classes, let, beta reduction ...)
    - The rules in Spoofax's DSLs are called "syntax-directed rules". Always one rule per syntax construct. Basically it's pattern matching on the AST.
      Type syntax with the code examples.
    - What do we mean with "Stratego is the most general aspect of Spoofax"? It is correct, but could use some supporting arguments. Book: Term Rewriting and all that. Mention that term rewriters are Turing complete.
    - s/rule-based operational semantics/structual operational semantics/g. Check the DynSem paper to see what DynSem uses and for a DynSem description. Just cite it.
    - A weakness of this section is that the code examples touch many concepts that are not explaind. That makes sense since it's out of the report's scope, though.
- REPL section (section 2):
    - Needs an explanation of a REPL's execution model (define it) in comparison with a usual compilation cycle. Gerlof mentioned this is not yet fully clear to ourselves with questions such as "does every expression need to be a program"? Skip replied: "it seems not, because there is already an environment of previous expressions". Hendrik agreed.
    - "such functionality" is unclear.
    - The section on homoiconicity is interesting, but its purpose is not entirely clear. What point are we trying to make?
    - "several advantages", but we only give one. Either make it "this is the main point" or mention more.
    - The input/output history paragraph confuses state and history. Untangle these two concepts.
    - The multiline editor paragraph should say instead "many languages have natural multiline constructs". The code example can go.
    - Keep it purely descriptive: right now the requirements show.
    - The code completion paragraph mentions context-sensitive completion and TAB completion. These are better known as semantic and syntactic completion respectively. Semantic completion is not distinct but rather a restriction.
    - The help and documentation paragraph: it's not descriptive about what has been studied.
    - Explicitly link the matrix to the paragraphs.
- Literate Programming section (section 3):
    - Make the connection between a REPL and a notebook and don't explain the execution model again.
    - IPython is more a REPL than WEB: IPython *is* a REPL with Jupyter UI on top.
    - Execution model: is it an incrementally built up program? Can one do things here that they

cannot in a REPL?

- Problem Definition section (section 4):
  - Something is off, but Hendrik doesn't know yet what. An email will follow.
- Problem Analysis section (section 5):
  - Mentions a "program", but a REPL does not have that.
  - Namebinding part is a little unclear because the execution model is still undefined. Also, replacing AST sections sounds very cool but Hendrik doesn't know if it's feasable (from our explanation).
  - We are allowed to require language developers to define certain things. An unanswered question is if someone defines the language and another person can then define the REPLS, or if this needs to be the same person.
- Requirements Analysis section (section 6):
  - Paragraphs vs. subsubsections.
  - Language-agnostic: is that possible in regards to previously mentioned issues? Perhaps this should be mentioned earlier: we need to find a balance between being language-agnostic and useful to a language. For example, we could say "One does not need to change the language, but for more features in the REPL there is the possibility to extand the language description".
  - Code completion: can Spoofax do this? Probably only syntactic but not (yet?) semantic. Do not require something from ourselves that is not (yet) offerend in Spoofax.
  - Automatic bind to variable: this is a new concept introduced suddenly. It has to link back to a previous section.
    It does, but because of the confusion between state and history (see the REPL section above), this was not clear.
  - Consistently use "REPL" (once we use "shell").
- Development Tools section (section 7):
  - Slightly more Project Plan related as opposed to research, but it is good to make the connection. Do verify the same is mentioned in the project plan.