Anna Jonsson
aj@cs.umu.se

Umeå University
June 6, 2016

Project log

This document contains the project log for my master thesis project (*5DV097: Examensarbete för civilingenjörsexamen i teknisk datavetenskap, 30.0 hp*).

January 7 Today, I had a meeting with Frank about what to incorporate in the thesis as well as how to write a thesis proposal, which I also did.

January 8 Thesis proposal sent in (after incorporating Frank's feedback) and accepted!

January 21 After finishing *Student conference in computing science* with the actual conference last week and having got started on *Parallel programming for multicore-based systems* this week, I have today evening tried to make a project plan for the thesis work.

January 22 Met with Frank, we decided on a project plan, so now I can finally start working for real.

January 25 Drew the boy-girl grammar by hand and by that learnt the syntax for the grammar file. Also made some sketches over how the grammar could be modified. One thing I found curious is that the grammar can generate disconnected graphs. A (quite sad) example of this follows.

```
(w1 / want :arg0 (g1 / girl) :arg1 g1)
(b1 / boy)
```

I will look into this a bit more, and if it should not be allowed, I may have to rewrite the part of the grammar that assigns relations as well.

January 26 Working on grammar. Some thoughts:

- We just need at maximum one leaf node each of `want` and `believe` as they represent the same event. For example, as is the case with `b1` in the following

```
(w1 / want :arg0 (s / she) :arg1 (b1 / believe :arg0 s :arg1 s))
(w2 / want :arg0 (h / he) :arg1 b1)
```

("She wants to believe herself and he wants her to believe herself") as opposed to this case where we have to define a new event (`b2`):

```
(w1 / want :arg0 (s / she) :arg1 (b1 / believe :arg0 s :arg1 s))
(w2 / want :arg0 (h / he) :arg1 (b2 / believe :arg0 h :arg1 s))
```

("She wants to believe herself and he wants to believe her").

- If we want to build only connected AMRs, we need to make sure that each leaf has at least one edge connected to it (which is not done at the moment).
- The concepts `want` and `believe` can only be connected with edges labelled `arg1`.
- The number of edges to the concepts `want` or `believe` is limited by the number of roots in the AMR.
- Thus, an AMR with neither `boy` or `girl` cannot have any edges labelled `arg0`. Furthermore, this type of AMR cannot have both `want` and `believe` as leaves and be connected at the same time (since only one `arg1` relation is allowed for each concept, which gives us a tree).

January 27 Grammar which allows `want` and `believe` as leaves done, but when trying to parse it, it failed. Looking deeper into it, I eventually found a quite well-hidden error message: "Can NOT create a parser for grammar WantBelieve2. Reason: Rules with lhs Z are not separated." I traced this back to the source code file `java.Contextual_Hyperedge_Replacement/src/generator/src/de/unibwm/inf2/ptd/main/Analysis.java`, and more specifically, `java.Contextual_Hyperedge_Replacement/src/generator/src/de/unibwm/inf2/ptd/analysis/RuleChoiceAnalysis.java` since I did not understand what is meant by "separated". I will try to see if there is a way to fix the grammar. In any case, I must understand why the current version does not work.

January 28 Tried a smaller grammar that uses only `want` and `believe` which I could not immediately get to work, but with a bit of effort and some PTD parser paper reading, it worked. Apparently, the problems had to do with the order in which the right-hand side parts of the rules were defined. Having tried to rewrite the larger grammar in similar ways and still not getting rid of the error message "Can NOT create a parser for grammar WantBelieve2. Reason: Rules with lhs Z are not separated." is not a good sign, but I will make another effort or two before giving up on it entirely.

January 30 Went deeper into the code and added some additional prints for the errors with the following result:

```
Can NOT create a parser for grammar WantBelieve2
Reason: Rules with lhs Z are not separated:
* init1 and init2 have common nh:
[arg2(-,-), believe(-), boy(-), girl(-), stmt(-), 2*want(-), ?]
* init1 and init3 have common nh:
[arg2(-,-), 2*believe(-), boy(-), girl(-), stmt(-), 2*want(-), ?]
* init1 and init6 have common nh:
[arg2(-,-), believe(-), boy(-), girl(-), stmt(-), 2*want(-), ?]
* init2 and init3 have common nh:
[arg2(-,-), 2*believe(-), boy(-), girl(-), stmt(-), want(-), ?]
* init2 and init6 have common nh:
[arg2(-,-), believe(-), boy(-), girl(-), stmt(-), want(-), ?]
* init3 and init6 have common nh:
[arg2(-,-), 2*believe(-), boy(-), girl(-), stmt(-), ?]
* init4 and init7 have common nh:
[arg2(-,-), believe(-), boy(-), stmt(-), 2*want(-), ?]
* init4 and init8 have common nh:
[arg2(-,-), 2*believe(-), boy(-), stmt(-), 2*want(-), ?]
* init4 and init11 have common nh:
[arg2(-,-), believe(-), boy(-), stmt(-), 2*want(-), ?]
* init5 and init9 have common nh:
[arg1(-,-), believe(-), girl(-), stmt(-), 2*want(-), ?]
* init5 and init10 have common nh:
[arg1(-,-), believe(-), girl(-), stmt(-), 2*want(-), ?]
* init5 and init12 have common nh:
[arg1(-,-), believe(-), girl(-), stmt(-), 2*want(-), ?]
* init7 and init8 have common nh:
[arg2(-,-), 2*believe(-), boy(-), stmt(-), 2*want(-), ?]
* init7 and init11 have common nh:
[arg2(-,-), boy(-), stmt(-), 2*want(-), ?]
* init8 and init11 have common nh:
[arg2(-,-), 2*believe(-), boy(-), stmt(-), want(-), ?]
* init9 and init10 have common nh:
[arg1(-,-), believe(-), girl(-), stmt(-), want(-), ?]
* init9 and init12 have common nh:
[arg1(-,-), believe(-), girl(-), stmt(-), want(-), ?]
* init10 and init12 have common nh:
[arg1(-,-), believe(-), girl(-), stmt(-), want(-), ?]
* init13 and init14 have common nh:
[arg2(-,-), believe(-), stmt(-), want(-), ?]
```

That is, the problem is that the parser cannot (for this grammar), for exam-

ple, differentiate between an AMR that is to have both `want` and `believe` as leaves and an AMR that has only `believe` as a leaf but `want` as its parent. To solve this one would have to find another way to build the AMRs such that the grammar is in a sense non-ambiguous all the way. I believe that it is in the paper expressed more or less as "if you at some point in the derivation can use two different rules, the sets of rest graphs (remainders to parse at that particular derivation step) of all possible input graphs must be disjoint". When altering the new grammar such that the leaf `want` and `believe` were defined as two terminal labels of their own, the grammar could be parsed (but I could not tell if the entire corpus was generated by it since I would have to alter the names of the leaf nodes in all 10000 AMRs. Besides, this should not have to be done. At least it generated the same 8880 AMRs as before).

Also, I found that the AMR

```
(w1 / want :arg0 (s / she) :arg1 (b1 / believe :arg0 s :arg1 s))
(w2 / want :arg0 (h / he) :arg1 b1)
```

does not parse even with the original grammar since `b1` is not recognised when it is referred to the second time. Thus, one would have to define an entirely new `believe` and connect `s` to it in the same way, but this seems to ruin the idea of AMRs by tree-ifying the graphs.

Things I want to try:

1. Experiment with the small grammar with only `want` and `believe` and see whether or not the "order" in which the relations are added matters for the rules to be separated
2. If the relation "order" seemed to have an impact of the separation of the rules, change it for the new grammar as well
3. Remove the AND part of the original grammar (it is not used in the corpus anyways) to remove the tree-ification problem (rather: remove the necessity to solve that problem for now) and make sure that the new grammar still parses 8880 of the AMRs of the corpus
4. Remove AND part of the new grammar as well and see if that simplifies things

February 4 When trying to alter the relation adding "order", the error "grammar does not support free edge choice" occurred. Removing the possibility to have several root nodes still parses 8880 boy-girl AMRs (as expected), but it does not seem to make the grammar any "easier". I believe that the key for solving this is to be able to handle (and not get rid of) cases such as

```
(w1 / want :arg0 (s / she) :arg1 (b1 / believe :arg0 s :arg1 s))
(w2 / want :arg0 (h / he) :arg1 b1),
```

which is (as already stated above) not parsed at the moment. This does not seem entirely trivial, but tomorrow is the biweekly meeting with Frank and I will bring this up for discussion then.

February 5 Meeting with Frank. Discussed PTD parsability and why some grammars are (not) PTD parsable. Decided to try another approach when building the grammars, namely to think in terms of machine learning for building AMR grammars, which would probably mean considering each concept (with possible [incoming/outgoing] relations) separately. Since I have not yet succeeded in building a PTD parsable boy-girl grammar that generates the entire boy-girl corpus, we decided that I should try to write the proof for the already existing grammar instead.

March 4 Discussed changes of project plan on the Frank meeting. We decided just to keep it as it is order-wise and go back to it when I have finished the course in parallel programming that has been taking up all of my time recently.

March 24 This week I have been modifying the report template and started to write the report in the form of a minimal outline of the correctness proof for the boy-girl grammar. In order to do this, I have read up on correctness proofs for grammars. The (short) week was ended by a meeting with Frank where the proof idea was discussed and gave me a lot more to work with.

April 30 Have now prepared an outline of the proof. Although, I have to build the grammar as well in order to be able to refer to the rules in the proof.

April 3 Two different versions of the grammar are now finished. The second version was made as a bit smaller version that would supposedly make referring to certain recurrence in the rules easier. Moreover, I have some questions regarding how to actually present the grammar. Was planning on improving the proof draft, but building the grammars using tikz took a lot more time than planned.

April 4 Frank-meeting. All my grammar drawing questions were answered, and we will discuss the proof later this week (since I clumsily forgot to send the draft in advance). This week I will work on writing about the work that was done to extend the boy-girl HRG to include AMRs with `want` and `believe` as leaves as well as continuing the proof-writing.

April 6 Another Frank-meeting. Discussed how to define the boy-girl language formally and how to construct a proof by induction using the properties formulated for each nonterminal. Also, Frank told me about some more powerful HRGs that could be interesting for AMR generation.

April 11 When working on writing down the formal definition of the boy-girl language (and the proof), I realised that it gets a bit weird that the grammar can generate disconnected graphs with at most two components (one containing `girl` and the other containing `boy`), and at most one of the components can be a single `boy` or `girl`. When generating a connected graph, though, it is not possible generate any of the two one-node graphs. This makes the language more difficult to express (and the proof harder?).

April 13 Meeting: a simple addition to the formal definition fixes the issue mentioned in the previous entry. This along with some discussion about whether or not it actually makes sense to allow a disconnected graph with one component as a single `boy` or `girl` node will be added. Next is to properly specify/define AMRs and HRGs in order to make sure that the correct framework is used when constructing the proof.

April 21 Done with the definitions of directed graphs, hypergraphs, AMRs and HRGs (for now). Added the extended grammar to the report. Have started to write the proof using the current definitions. The things I have found difficult (and that I want to discuss during the next meeting) are:

- Separating the induction hypothesis from the "new" information.
- Q: Should it not be made sure that *none* of the conditions are violated for each nonterminal?
- Q2: ... Or should just the non-trivial conditions be picked out and handled?
- Using the components derivated from each nonterminal in the induction step rather than just saying "performing another derivation step is fine".

April 22 Today's meeting was mostly about me having to define everything a lot more in order to be able to express the proof in a sensible way. Other than that, we talked about the proof itself, and I got a better idea of how it should be formulated.

April 26 Although having spent the greatest part of the last two days on an application for a Ph D position, I have managed to make some progress on the project, not in the least in my brain in the form of a vision of how the resulting definitions and the like will look like, which I lacked previously. Next time we will discuss how the remaining time is best spent and possibly make some rearrangements in the plan.

April 28 The definitions are rewritten (definitely worth the rather long time spent on it) which made the proof so much easier to start on, but I am however not satisfied at all with the current version of the first implication of the proof. For the meeting tomorrow, I want to discuss: time plan, the second implication of the proof, and something little about why the extended grammar is not PTD parsable.

April 29 Meeting. Discussed the things mentioned in the previous entry, and I really hope that I will be able to finish everything in time. We agreed that I would update the report continuously (even though I will primarily work on reading about/building grammars of the restricted type) so that Frank can read and give feedback every week.

May 3 Errors for the extended boy-girl grammar using the most recent version of the PTD parser:

```
Could not create a parser for grammar WantBelieve2
Reasons:
1) Rules with lhs Z are not separated
2) Rules with lhs Z are not separated:
* init2 and init1 have common nh:
[arg1(-,-), arg2(-,-), 2*believe(-), boy(-), girl(-), stmt(-), want(-), ?]
3) Rules with lhs Z are not separated:
* init3 and init1 have common nh:
[arg1(-,-), arg2(-,-), 2*believe(-), boy(-), girl(-), stmt(-), want(-), ?]
4) Rules with lhs Z are not separated:
* init3 and init2 have common nh:
[arg2(-,-), believe(-), boy(-), girl(-), stmt(-), want(-), ?]
5) Rules with lhs Z are not separated:
* init6 and init1 have common nh:
```

```
[arg1(-,-), arg2(-,-), 2*believe(-), boy(-), girl(-), stmt(-), want(-), ?]
6) Rules with lhs Z are not separated:
* init6 and init2 have common nh:
[arg2(-,-), believe(-), boy(-), girl(-), stmt(-), want(-), ?]
7) Rules with lhs Z are not separated:
* init6 and init3 have common nh:
[arg2(-,-), believe(-), boy(-), girl(-), stmt(-), want(-), ?]
8) Rules with lhs Z are not separated:
* init7 and init4 have common nh:
[arg2(-,-), 2*believe(-), boy(-), stmt(-), want(-), ?]
9) Rules with lhs Z are not separated:
* init8 and init4 have common nh:
[arg2(-,-), 2*believe(-), boy(-), stmt(-), want(-), ?]
10) Rules with lhs Z are not separated:
* init8 and init7 have common nh:
[arg2(-,-), 2*believe(-), boy(-), stmt(-), want(-), ?]
11) Rules with lhs Z are not separated:
* init9 and init5 have common nh:
[arg1(-,-), 2*believe(-), girl(-), stmt(-), want(-), ?]
12) Rules with lhs Z are not separated:
* init10 and init5 have common nh:
[arg1(-,-), 2*believe(-), girl(-), stmt(-), want(-), ?]
13) Rules with lhs Z are not separated:
* init10 and init9 have common nh:
[arg1(-,-), believe(-), girl(-), stmt(-), want(-), ?]
14) Rules with lhs Z are not separated:
* init11 and init4 have common nh:
[arg2(-,-), 2*believe(-), boy(-), stmt(-), want(-), ?]
15) Rules with lhs Z are not separated:
* init11 and init7 have common nh:
[arg2(-,-), believe(-), boy(-), stmt(-), want(-), ?]
16) Rules with lhs Z are not separated:
* init11 and init8 have common nh:
[arg2(-,-), 2*believe(-), boy(-), stmt(-), ?]
17) Rules with lhs Z are not separated:
* init12 and init5 have common nh:
[arg1(-,-), 2*believe(-), girl(-), stmt(-), want(-), ?]
18) Rules with lhs Z are not separated:
* init12 and init9 have common nh:
[arg1(-,-), believe(-), girl(-), stmt(-), want(-), ?]
19) Rules with lhs Z are not separated:
* init12 and init10 have common nh:
[arg1(-,-), believe(-), girl(-), stmt(-), want(-), ?]
20) Rules with lhs Z are not separated:
```

```
* init14 and init13 have common nh:
[arg2(-,-), believe(-), stmt(-), want(-), ?]
```

The unseparated edges are the same (common nh varies, but that must be because the analysis seems to be done in a different order this time?).

May 5 Finished a draft of the second implication of the proof as well as a definition of rDAGs. Next: construct a boy-girl rDAG.

May 6 Meeting during which I learnt and we discussed why the extended boy-girl grammar is not PTD parsable. Basically, it has to do with the PTD analysis first separating the rules, and then identifying the start nodes, which means that the information about which nodes are leaf nodes (which one could intuitively use to determine uniquely which rule to apply next) is lost. Moreover, some fixes of the second implication of the proof has to be done as well as a total restructuring of the report. After implementing this, building an rDAG must be done.

May 8 Now, the proof should really be done, but I guess I will have to revise it at least once more. The report has been restructured, and figures of some of the rules in the extended boy-girl grammar (the ones that have to be added in the case of all four types of leaf nodes) have been added. Describing one of the cases a bit more thoroughly should suffice for the reader to be convinced, I hope. Tomorrow, I can finally start building the rDAG (I hope it will not rain).

May 10 Working on completing the grammar and parsing chapters. Added some explanations about context-freeness to the section where the context-freeness lemma is presented (I wonder whether it should go in the HRG section instead). It is also hard to know what must be in the discussion section and what goes in the result-containing ones.

May 12 During the last two days I have worked on building an rDAG boy-girl grammar (with Slack support from Frank) and including it into the report. However, I always find new things to fix — need to structure my intra-brain search tree.

May 13 Meeting: got a print-out of my thesis full of great comments. Discussed how to structure the last parts and what to add.

May 16 All of the previously received comments have now been handled, and today I received another set of feedback (this time for the rDAG chapter) and took care of them. Also, I wrote a (bad) abstract. The plan is now to write the discussion and the conclusion and future work chapters. Finally, I have to add things to the introduction, and then, the only thing remaining will be reading through the entire thing to make sure that it is coherent, cohesive and correct (huehue).

May 17 Wrote outline (as part of the introduction), discussion, conclusion and future work and fixed abstract, introduction and problem description and sent it all to poor Frank.

May 18 Already got feedback, and started to make fixes accordingly.

May 19 Finished the final set of alterations and sent out a final draft.

May 20 Submitted the report.

May 23 Started working on the presentation.

May 25 Finished a draft of the presentation slides and received feedback on them from Frank. Test run of unfinished presentation took about 20 minutes.

May 27 Short meeting with Frank about the presentation and such. Received the presentation schedule and Viktor's report (the report I am to write an opposition report on) yesterday.

May 30 Implemented changes in presentation slides and started to read Viktor's report. When doing a couple of test runs of the presentation, it took 23-26 minutes.

June 1 This week, I have been preparing the opposition, and today, I submitted the opposition report (after the always helpful Frank had had a look on it). I have also gone through the presentation a number of times, and the runs seem to take between 22 and 26 minutes (I really hope that I will not exceed the limit of 25 minutes). Tomorrow: presentation. Help!

June 2 I survived the presentation!

June 5 Updated report according to the comments received from Viktor.

June 6 Submitted the final version of the report to Henrik after implementing some nice changes that were suggested by Frank.