Description of the Test Suite

Each directory contains one test. It contains a test.racer file, which is the actual knowledge base. It contains a test_query.racer file, which contains the query that was executed.

My test results were obtained on the following way:

- I started RACER once.
- Each test case connected to RACER through JRacer library.
- Each test loaded the knowledge base first and then executed the query. In this way I was sure that nothing was cached.
- The test executed the query using JRacer and has read all results. I measured the time that elapsed from the moment I invoked JRacer until all results were read.

I assume that reloading the file every time isn't really fair, since probably the first question always makes RACER do more work. If you could tell me what would be a more appropriate way of testing, I'd be mostly appreciative.

In general, it seems to me that it makes sense to cache the work done on the TBox – after all, TBox will not change that often. However, I'd like to switch off all caches of the work on the ABox, especially if these caches are cleared each time I change the ABox.

- This might be a possible answer: first as some question, then assert and delete some ABox assertion, then ask the question again. Do you think this would characterize the performance in a nicer way?

Description of Tests

symmetric_concept_tree_d_sc_i

The knowledge base is a symmetric tree of concepts. The tree has depth d. Each node in the tree is a concept having sc subconcepts and i instances. Subconcepts are directly classified under their parents using $SC \subseteq C$.

Query involved reading the extension of some concept at the top of the tree.

symmetric_concept_tree_with_relations_d_sc_i

The same as the previous test. However, the knowledge base contains tons of role assertions. However, no role is mentioned at all in any of the concept definitions. Hence, I didn't expect a slowdown. However, it was surprised to see that RACER was significantly slower in this case.

symmetric_concept_tree_with_relations2_d_sc_i

Same as the previous test, but with smaller number of different roles. However, the total number of role assertions is the same. I wanted to see whether the number of different roles makes a difference, or if the slowdown is due to the number of role assertions.

disjunction_concept_tree_with_relations_d_sc_i_r

The knowledge base is a symmetric tree of concepts. There is the total of r roles. The tree has depth d. Each concept has i instances, where instances are linked by the roles on the round-robbin basis (each next generated instance is linked to the previously generated instance through one role). Each concept is defined as $\exists R.T \cup SC \subseteq C$.

Hence, the knowledge base is not really disjunctive, but simulates something that can be done in the Horn fragment.

disjunctive_concept_tree_ c_i

Also a symmetric tree with i instances per concept. Each concept is defined as $SC \subseteq C_1 \cup C_2$. In this way an extension of any concept consists of exactly those instances which are directly classified under the concept. However, the disjunction makes inference engine do a lot of work to figure this out.

kaon_ontology

This is a conversion of one of our ontologies which we used on one project. As far as I remember, this ontology doesn't use any of the advanced stuff.

siemens_extended_ontology

This is the previous ontology, but extended with randomly generated instances and with some disjunction.