Some ideas for TPTP

Josef Urban

Formula types

General remark for Geoff: make the BNF syntax hyperlinked (I can do it if necessary). As it will grow longer, it will be even more painful to find something there.

I don't know what is meant by the *hypothesis* formula type, and when should *lemma*, *theorem* and *plain* be used. Comments specifying this should be in the syntax. Formula types seem to overlap a bit with their inference status - e.g. E prover puts status *thm* for almost everything derived. I am not sure how to encode the Mizar natural-deduction proofs using these types. For now I most often use *lemma-derived* (E prover uses *plain-derived* most often) for everything that is proved and not a Mizar theorem. I also use *theorem*, *definition*, *sort* and *unknown*. However *sort* formulas are also theorems or definitions in Mizar. I'll probably also need the *assumption* type (which E prover already uses for clausification). It is not in the syntax yet.

Syntax for models

Our group plans to work with (pseudo)models for knowledge-based ATP in some short time, but I have not worked with MACE or similar model generators yet. I looked at Koen's and Geoff's proposal, and if possible, I'd suggest to extend it to be able to express infinite models. You can e.g. use an "exception-driven" syntax for specifying functor and predicate interpretations, i.e. some default value is specified for all domain elements, and if more specific definitional clause exists, it overrides the more general clause - this technique is used pretty often in programming.

XML for TPTP

XML is too verbose to read by humans, but it is very easy to transform to human-readable (e.g. Prolog) format (Petr Pudlak has written such tools for his XFF). XML has established and standardized mechanisms for syntax extension and validation, and a vast number of tools for anything imaginable. To put it simply, it has industry support. If TPTP was only about pure first-order formulas, the current Prolog syntax would be sufficient. But the number of TPTP extensions and annotations will probably grow (it is quite rich already now). I guess that if we continue to extend the Prolog syntax, we will eventually rediscover things like validation, extension specification, etc.

Random stuff

A TPTP compatibility layer for arithmetical symbols (mapped correctly to each prover which supports some arithmetic). Common TPTP tools for parsing the TPTP proofs, and some standard API (display proof tree, list axioms used, etc). More standardization of the *useful-info* slot. Could provers ensure parsing of *useful-info* for initial clauses, and let me specify how it is inherited during inferences, and the inference heuristics using it? Loading of large KBs. Sorted and higher-order extensions - see my ESCAR paper.