External Review of CASL Extensions

Recommendations and suggestions produced by the review committee, and reactions of the proponents to the reviews

The review committee put together by José Meseguer included the following additional members: Didier Bert, Rolf Hennicker, Tom Maibaum, Peter Padawitz, and Carolyn Talcott.

The committee met on March 30 at UPC in Barcelona for a one-day open meeting in which the five CASL extensions were presented and discussed.

The reviews were unanimously subscribed by all the members.

1. HAS-CASL

Recommendation: approve

Suggestions:

a. The authors should take into account the detailed comments by internal reviewers: even though a response had been given, several of those comments can still help improving the extension.

b. The relationship between Haskell programs and HAS-CASL specification and reasoning should be further clarified; also, some comparison with other Haskell logics would be appropriate.

Reactions of the proponents:

a. The comments by the internal reviewers had already been largely incorporated into the HasCASL summary; a mildly edited new version of the summary, which also takes into account the (minor) remaining points, is available on the HasCASL home page http://www.informatik.uni-bremen.de/agbkb/forschung/formal_methods/CoFI/HasCASL/

   Moreover, the construction which ensures the satisfaction condition for polymorphic HasCASL is now available as a technical report under http://www.informatik.uni-bremen.de/~lschrode/papers/genpoly.ps

   The details concerning conservativity of HasCASL over CASL and the semantics of sort generation constraints laid out in the quoted response to the reviewers reports will be incorporated in the forthcoming HasCASL semantics document. (The basic facts as such are mentioned also in the summary.)

b. The claim that HasCASL's program blocks translate trivially into Haskell has in the meantime been substantiated by the implementation of such a translation as part of the heterogeneous tool set hets. The detailed nature of the semantics thus induced for a sublanguage of Haskell is future work, as is the comparison between HasCASL and the program logic for Haskell used in the Programatica project.

2. CO-CASL

Recommendation: approve

Suggestions:

The existing document could benefit from further explanations and examples. In particular, some of the questions raised by members of the committee and the answers provided by the proposers could provide a good way to add those additional explanations, for example about: binary observers, specifying finite and infinite lists, modal formulas axiomatising substructures, and so on.

Reactions from the proponents:

The suggestions will be incorporated in a new version of the document in question, which is presently in preparation; in particular, there will be improved example material providing in particular a better illustration of the use of modal logic.

3. CSP-CASL

Recommendation:

The proposal is work in progress; it could be approved, provided the missing work is done. It needs another round of internal CASL review.

Suggestions:

This is not the only combination between an algebraic specification language and a process algebra formalism, LOTOS being perhaps the most well-known example. It would be helpful to explain more fully the rationale behind this choice, the relationships with other such combinations, and the special advantages of the choice in question.

Reactions of the proponents:

I would like to thank the referees as well as the External Review Committee for reviewing CSP-CASL and providing me with thorough feedback.

Work on CSP-CASL continues in three directions:

a. further theoretical studies on the integration of data and processes;

b. tool-development; and

c. establishing a collection of significant specifications (there is already an industrial case study on its way).

This will help to comment on the reviewers' suggestions to better explain the rationale behind CSP-CASL, relate it on a deeper level with other approaches, and point out the special advantages of CSP-CASL's way to integrate data and processes. With convincing answers to these points, I intend to re-submit CSP-CASL for approval as an official CASL extension.

4. LTL-CASL

Recommendation: approve

Suggestions:

a. If at all possible, the LTL acronym should be replaced by a less confusing one (like LTS) to avoid the obvious confusion with linear time temporal logic for which the LTL acronym is universally used.

b. It would very much help this extension to clarify the exact relationship with CTL*, perhaps as a map of logics or institutions. This could have the great advantage of endowing this extension with the excellent CTL* tool support, since currently no tools exist for the extension.

c. It would also be quite helpful to develop in more detail the semantic relationships with CO-CASL models, both at the level of models, and at the level of the modal/temporal logics describing properties.

Reactions of the proponents:

a. It seems that it would be less confusing to use CASL-LT, so as to avoid the name clash with LTL, and still stay close to what was used before (while CASL-LTS might be thought of as a brand new name for something new).

b. It is clear that there are some common features with CTL*. We could describe what the restrictions/transformations needed so as to be able to benefit from CTL* tool support.

c. This interesting issue can be dealt with in collaboration with colleagues working on CO-CASL and/or semantics.

5. HET-CASL

Recommendation: approve

Suggestions:

a. Since this is a meta-framework to combine heterogeneous formalisms, it appears that the authors could incorporate in it new formalisms as they come along, including combinations corresponding to CASL extensions.

b. It would be very helpful if the authors were to give interesting examples of heterogeneous proof obligations and proofs.

Reactions of the proponents:

a. Of course. Let me add some remarks on what authors of a new language could/should do.

Note that HetCASL is generic over a logic graph, hence there the incorporation of new languages does not change anything what HetCASL concerns. The only requirements are

- that the new language come as an institution with fully qualified symbols,
- that the new language is be related to (a sublogic of) CASL (if possible) via some institution (co)morphism.

The responsibility to show that these requirements are fulfilled should be with the language proposers.

Another point is the integration of new languages into the heterogeneous tool set (Hets). Here, the Hets team can assist language designers with the development of analysis tools for their language and their integration into Hets.

b. See

http://www.brics.dk/~pdm/IFIP-WG1.3/Meeting-2002-Chiemsee-Abstracts.html
http://www.informatik.uni-bremen.de/~till/papers/ifip02.pdf
http://www.informatik.uni-bremen.de/~till/papers/ifip02-paper.pdf

for a toy example of a heterogeneous proof.

I am working on more realistic examples.