Acta Cybernetica **14** (1999) 1-.

Extending the Sparkle Core language with object abstraction [*]

Máté Tejfel, Zoltán Horváth, Tamás Kozsik[†]

Abstract

Sparkle is a theorem prover specially constructed for the functional programming language Clean. In a pure functional language like Clean the values of the functional variables are constants; variables of functional programs do not change in time. Hence it seems that temporality has no meaning in functional programs. However, in certain cases (e.g. in interactive or distributed programs, or in ones that use IO), we can consider a series of values computed from each other as different states of the same "abstract object". For this abstract object we can prove temporal properties. In this paper we present a method to describe abstract objects and temporal properties in an extended version of the Sparkle Core language. The creation of such descriptions will be supported by a refactoring tool. The descriptions are completely machine processible, and provide a way to automatize the proof of temporal properties of Clean programs with the extended Sparkle system.

Categories and Subject Descriptors: D.1.1 [Programming Techniques]: Applicative (Functional) Programming; F.3.1 [Logics and meanings of programs]: Specifying and Verifying and Reasoning about Programs - *invariants*;

Key Words and Phrases: Verification, invariant properties, abstract functional object, Clean, Sparkle

References

[1] Achten, P., Plasmeijer, R.: Interactive Objects in Clean. *Proceedings of Implementation of Functional Languages, 9th International Workshop, IFL'97* (K. Hammond et al (eds)), St. Andrews, Scotland, UK, September 1997, LNCS 1467, pp. 304–321.

[2] Butterfield, A., Dowse, M., Strong, G.: Proving Make Correct: IO Proofs in Haskell and Clean. *Proceedings of Implementation of Functional Programming Languages*, Madrid, 2002. pp. 330–339.

[*]Supported by the Hungarian National Science Research Grant (OTKA), Grant Nr.T037742. and by the Bolyai Research Scholarship.

[†]Department of Programming Languages and Compilers Eötvös Loránd University, Budapest, e-mail: `matej@inf.elte.hu, hz@inf.elte.hu, kto@inf.elte.hu`

[3] Butterfield, Andrew: Reasoning about I/O and Exceptions. *Proceedings of Implementation and Application of Functional Languages, IFL'04*, Lbeck, September 8-10, 2004., pp. 33-48.

[4] Chandy, K. M., Misra, J.: *Parallel program design: a foundation*. Addison-Wesley, 1989.

[5] Dam, M., Fredlund, L., Gurov, D.: Toward Parametric Verification of Open Distributed Systems. *Compositionality: The Significant Difference* (H. Langmaack, A. Pnueli, W.-P. De Roever (eds)), Springer-Verlag 1998.

[6] Dijkstra, E. W.: *A Discipline of Programming*. Prentice-Hall Inc., Englewood Cliffs (N.Y.), 1976.

[7] Diviánszky P. - Szabó-Nacsa R. - Horváth Z.: A Framework for Refactoring Clean Programs. *6th International Conference on Applied Informatics*, Eger, Hungary January 27-31 2004.

[8] Dowse, M., Butterfield, A., van Eekelen, M., de Mol, M., Plasmeijer, R.: Towards Machine-Verified Proofs for I/O *Proceedings of Implementation and Application of Functional Languages, IFL'04*, Lbeck, September 8-10, 2004., pp. 469-480.

[9] Dowse, M., Butterfield, A.: A Language for Reasoning about Concurrent Functional I/O (Draft) *Proceedings of Implementation and Application of Functional Languages, IFL'04*, Lbeck, September 8-10, 2004., pp. 129-141.

[10] Horváth Z.: The Formal Specification of a Problem Solved by a Parallel Program—a Relational Model. *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae, Sectio Computatorica*, Tomus XVII. (1998) pp. 173–191.

[11] Horváth Z., Achten, P., Kozsik T., Plasmeijer, R.: Proving the Temporal Properties of the Unique World. *Proceedings of the Sixth Symposium on Programming Languages and Software Tools*, Tallin, Estonia, August 1999. pp. 113–125.

[12] Horváth Z., Achten, P., Kozsik T., Plasmeijer, R.: Verification of the Temporal Properties of Dynamic Clean Processes. *Proceedings of Implementation of Functional Languages, IFL'99*, Lochem, The Netherlands, Sept. 7–10, 1999. pp. 203–218.

[13] Horváth Z. - Kozsik T. - Tejfel M.: Proving Invariants of Functional Programs. *Proceedings of Eighth Symposium on Programming Languages and Software Tools*, Kuopio, Finland, June 17-18, 2003., pp. 115-126

[14] Horváth Z. - Kozsik T. - Tejfel M.: Verifying invariants of abstract functional objects - a case study. *6th International Conference on Applied Informatics*, Eger, Hungary January 27-31 2004.

[15] Home of Clean. http://www.cs.kun.nl/˜clean/

[16] Kozsik T., van Arkel, D., Plasmeijer, R.: Subtyping with Strengthening Type Invariants. *Proceedings of the 12th International Workshop on Implementation of Functional Languages* (M. Mohnen, P. Koopman (eds)), Aachener Informatik-Berichte, Aachen, Germany, September 2000. pp. 315–330.

[17] Kozsik T.: Reasoning with Sparkle: a case study. *Technical Report*, Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary.

[18] de Mol, Maarten. PhD thesis (in preparation), Radboud University Nijmegen.

[19] de Mol, M., van Eekelen, M., Plasmeijer, R.: Theorem Proving for Functional Programmers, Sparkle: A Functional Theorem Prover, Springer Verlag, LNCS 2312, p. 55 ff., 2001.

[20] Peyton Jones, S., Hughes, J., et al. *Report on the Programming Language Haskell 98, A Non-strict, Purely Functional Language*, February 1999.

[21] Plasmeijer, R., van Eekelen, M.: *Concurrent Clean Version 2.0 Language Report*, 2001. http://www.cs.kun.nl/˜clean/Manuals/manuals.html

[22] Szabó-Nacsa R., Diviánszky P., Horváth Z.: An Environment for Safe Refactoring Clean Programs. *CSCS 2004, The Fourth Conference of PhD Students in Computer Science*, Szeged, Hungary, July 1-4, 2004.