se puede escribir equivalentemente, como el problema de verificar el valor de verdad de la precondición $wp(S, true)$ para valores iniciales de las variables y constantes del programa S, es decir, para no contradecir la indecidibilidad del problema de la parada, debe ocurrir que verificar el valor de verdad de la aserción $wp(S, true)$, no es decidible en general.

V. Conclusiones

El teorema de decidibilidad del cálculo de wp provee una alternativa para calcular precondiciones más débiles de forma automática, sin embargo estas precondiciones, en muchos casos, no serían una aserción decidible. Por esta razón el teorema sugiere una aplicación de software para el cálculo de wp que maneje aserciones no decidibles de forma simbólica.

La idea práctica es usar todas las técnicas conocidas para calcular precondiciones o invariantes, y en caso de que estas técnicas fallen en el cálculo de una aserción decidible, la aplicación arroja como última opción, la aserción resultante del teorema de decidibilidad aquí mostrado.

Una aplicación de este estilo trabajaría en muchas ocasiones con aserciones no decidibles, por lo que no tiene sentido manejar estas aserciones dentro del lenguaje de programación en cuestión, porque en estos casos las aserciones no serían programables. Por esta razón sugiero que una aplicación que haga uso del teorema de decibilidad aquí mostrado, maneje las aserciones como comentarios al código. Por ejemplo se pudiera desarrollar un IDE que inserte a modo de comentario, de forma automática la precondición más débil de cada instrucción. Un IDE de este tipo siempre puede computar una aserción entre todas las instrucciones del programa, aunque algunas de ellas sean no decidibles, pudiera ser provechoso, porque es posible que en el cálculo sucesivo de wp, las aserciones se simplifiquen y se obtenga, al final del proceso, una precondición más débil de todo el programa, que sea una aserción decidible.

Por otro lado desde el punto de vista teórico, los resultados aquí presentados muestran que la teoría de wp de Dijkstra es aplicable sobre la teoría de conjuntos, y por ende también el teorema de la invariancia y todas las reglas de Hoare derivadas de wp. De esta forma se pueden usar las técnicas de corrección formal sobre algoritmos con tipos de datos pertenencientes a la teoría de conjuntos. Por ejemplo, se puede usando wp o reglas de Hoare, corregir algoritmos donde los tipos de las variables son objetos como ordinales, cardinales, filtros, ultrafiltros, espacios topológicos, etc.

Referencias

[1] E. W. Dijkstra. *Guarded Commands, Nondeterminacy and Formal Derivation of Programs*. Communications of the ACM, 18(8):453457, 1975.

[2] D. Gries. *The Science of Programming*. New York, New York: Springer, 1981.

[3] F. Flaviani. *Modelo Relacional de la Teoría Axiomática del Lenguaje GCL de Dijkstra*. CoNCISa 2015, Valencia, Venezuela, Noviembre 2015, pp. 153-164.

[4] F. Flaviani. *Cálculo de Precondiciones Más Débiles*. ReVeCom, Diciembre 2016, Vol. 3, No. 2, pp. 68-80.

[5] F. Flaviani. *Calculation of Invariants Assertions*. CLEI, Septiembre 2017, http://www.clei2017-46jaiio.sadio.org.ar/sites/default/files/Mem/SLTC/sltc-04.pdf

[6] G. Winskel. *The Formal Semantics of Programming Languages: An Introduction*. MIT Press, 1993.

[7] J. Berdine, A. Chawdhary, B. Cook, D. Distefano, and P. OHearn. *Variance Analyses from Invariance Analyses*. Proceedings of the 34th Annual Symposium on Principles of Programming Languages, Nice, France, 2007.

[8] E. Rodriguez Carbonnell and D. Kapur. *Program Verification using Automatic Generation of Invariants*. International Conference on Theoretical Aspects of Computing, 2004, Vol. 3407, pp. 325340.

[9] J. Carette and R. Janicki. *Computing Properties of Numeric Iterative Programs by Symbolic Computation*. Fundamentae Informatica, 80(1-3):125146, March 2007.

[10] M. A. Colon, S. Sankaranarayana, and H. B. Sipma. *Linear Invariant Generation using non Linear Constraint Solving*. Computer Aided Verification, 2003, Vol. 2725, pp. 420432.

[11] M. D. Ernst, J. H Perkins, P. J. Guo, S. McCamant, C. Pacheco, M. S. Tschantz, and C. Xiao. *The Daikon System for Dynamic Detection of Likely Invariants*. Science of Computer Programming, 2006.

[12] J.C. Fu, F. B. Bastani, and I-L. Yen. *Automated Discovery of Loop Invariants for High Assurance Programs Synthesized using ai Planning Techniques*. HASE 2008: 11th High Assurance Systems Engineering Symposium, 2008, pp. 333342, Nanjing, China.

[13] L. Kovacs and T. Jebelean. *Automated Generation of Loop Invariants by Recurrence Solving in Theorema*. In D. Petcu, V. Negru, D. Zaharie, and T. Jebelean, editors, Proceedings of the 6th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC04), pages 451464, Timisoara, Romania, 2004. Mirton Publisher.

[14] L. Kovacs and T. Jebelean. *An Algorithm for Automated Generation of Invariants for Loops with Conditionals*. D. Petcu, editor, Proceedings of the Computer-Aided Verification on Information Systems Workshop (CAVIS 2005), 7th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2005), pages 1619, Department of Computer Science, West University of Timisoara, Romania, 2005.

[15] S. Sankaranarayana, H. B. Sipma, and Z. Manna. *Non Linear Loop Invariant Generation Using Groebner Bases*. In Proceedings, ACM SIGPLAN Principles of Programming Languages, POPL 2004, pages 381329, 2004.

[16] A. Gupta, A. Rybalchenko. *InvGen: An Efficient Invariant Generator*. International Conference on Computer Aided Verification, 2009, pp. 634-640.

[17] Stanford Invariant Generator, 2006, http://theory.stanford.edu/ srirams/Software/sting.html

[18] E. Rodrguez-Carbonell and D. Kapur. *Generating all Polynomial Invariants in Simple Loops*. J. Symbolic Comput. 42 (2007), no. 4, 443476.

[19] S. Magill, A. Nanevski, E. Clarke, and P. Lee. *Inferring Invariants in Separation Logic for Imperative List-processing Programs*. SPACE, 1(1):57, 2006.

[20] J. Berdine, B. Cook, and S. Ishtiaq, *SLAyer: Memory Safety for Systems-Level Code*. Gopalakrishnan, Springer, Heidelberg 6806:178183, 2011.

[21] C. Varming, L. Birkedal. *Higher-order Separation Logic in Isabelle/holcf*. Electronic Notes in Theoretical Computer Science, 218:371389, 2008.

[22] M. Barnett, K. Rustan, and M. Leino, Microsoft Research. *Weakest-Precondition of Unstructured Programs*. Proceedings of the 6th ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering, 31(2006):1, pp. 82-87.

[23] K. Kunen. *Set Theory*. College Publications, London, UK, 2013.