Proofs of Program Properties via Unfold/Fold Transformations of Constraint Logic Programs

Alberto Pettorossi[1], Maurizio Proietti[2], Valerio Senni[1]
(1) DISP, University of Roma Tor Vergata, Roma, Italy
(2) IASI-CNR, Roma, Italy

Abstract

In the literature there are various papers which illustrate the relationship between the unfold/fold program transformation techniques [1, 8] and the proofs of program properties both in the case of logic programs and in the case of functional programs (see, for instance,[2, 3, 5, 6, 7]). In this paper we will illustrate that relationship in the case of constraint logic programs. We build up on the results we have presented in [6] where we have considered logic programs with locally stratified negation. The constraint logic programming paradigm significantly extends the logic programming paradigm by allowing some of the atoms to denote constraints in a suitably chosen constraint domain D. By using those constraints it is often possible to get simple and direct formulations of problem solutions.

Constraint logic programming is very powerful and flexible because it is parametric w.r.t. the solver one may choose for the efficient solution of the constraints.

Analogously to what we have done in [6], here we present a technique for proving that a closed first order formula φ holds in the perfect D-model, denoted $M(P)$, of a stratified constraint logic program P. When this property holds we write $M(P) \models \varphi$.

Our proof method for showing that $M(P) \models \varphi$, consists of two steps:

Step 1: we introduce a statement $f \leftarrow \varphi$, where f is a new predicate symbol, and we use a variant of the Lloyd-Topor transformation [4] for transforming $f \leftarrow \varphi$ into a conjunction $F(f, \varphi)$ of clauses, possibly with negated atoms in their bodies, such that $P \wedge F(f, \varphi)$ is a stratified constraint logic program and $M(P) \models \varphi$ iff $M(P \wedge F(f,\varphi)) \models f$, and

Step 2: we show that $M(P \wedge F(f,\varphi)) \models f$ by applying semantics preserving transformation rules and deriving from $P \wedge F(f, \varphi)$ a new program of the form: $Q \wedge f$.

Indeed, we have that $M(Q \wedge f) \models f$ and, since the transformation rules are semantics preserving, at the end of Step 2 we have that $M(P \wedge F(f,\varphi)) \models f$ iff $M(Q \wedge f) \models f$.

Since f is false in the perfect D-model of any program which has no clause with head f, our method can also be used to show that $M(P) \not\models \varphi$, that is, $M(P) \models \neg\varphi$, by deriving from $P \wedge F(f, \varphi)$ a new program R, such that f does not occur in the head of any clause of R.

We will also present a program transformation strategy for applying our unfold/fold rules in a semi-automatic way. Our strategy may or may not terminate, depending on the initial program P and formula φ. However, we identify some classes of programs and formulas for which our strategy always terminates and, thus, our strategy is a decision procedure for checking whether or not $M(P) \models \varphi$ holds for any given program P and formula φ in those classes.

Dagstuhl Seminar Proceedings 05161
Transformation Techniques in Software Engineering
http://drops.dagstuhl.de/opus/volltexte/2006/422

References

[1] R. M. Burstall and J. Darlington. A transformation system for developing recursive programs. *Journal of the ACM*, 24(1):44–67, January 1977.

[2] B. Courcelle. Equivalences and transformations of regular systems – applications to recursive program schemes and grammars. *Theoretical Computer Science*, 42:1–122, 1986.

[3] L. Kott. Unfold/fold program transformation. In M. Nivat and J.C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 411–434. Cambridge University Press, 1985.

[4] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, Berlin, 1987. Second Edition.

[5] A. Pettorossi and M. Proietti. Synthesis and transformation of logic programs using unfold/fold proofs. *Journal of Logic Programming*, 41(2&3):197–230, 1999.

[6] A. Pettorossi and M. Proietti. Perfect model checking via unfold/fold transformations. In J. W. Lloyd, editor, *First International Conference on Computational Logic, CL 2000, London, UK, 24-28 July, 2000*, Lecture Notes in Artificial Intelligence 1861, pages 613–628. Springer, 2000.

[7] A. Roychoudhury, K. Narayan Kumar, C. R. Ramakrishnan, I. V. Ramakrishnan, and S. A. Smolka. Verification of parameterized systems using logic program transformations. In *Proceedings of the Sixth International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2000, Berlin, Germany*, Lecture Notes in Computer Science 1785, pages 172–187. Springer, 2000.

[8] H. Tamaki and T. Sato. Unfold/fold transformation of logic programs. In S.-Å. Tärnlund, editor, *Proceedings of the Second International Conference on Logic Programming*, pages 127–138, Uppsala, Sweden, 1984. Uppsala University.