When Linear Norms Are Not Enough

Nachum Dershowitz * Naomi Lindenstrauss † Yehoshua Sagiv†
Alexander Serebrenik†‡

We consider left termination of queries to logic programs ([Apt97, DSD94]). The general approach to proving termination is by showing that during the evaluation of a given program, there is a sequence of terms, such that the size of these terms is decreasing. Usually, orders induced by defining linear norms on terms ([UvG88, Plü90, LS97, CT97]) are sufficient for showing termination. However, in some cases, termination cannot be shown using a linear norm. Two examples are given below.

1. Consider the program for repeated derivations with a query that is symbolically represented in the form `d(Expression, Derivative)`, where the first argument is ground.

```
d(der(t), 1).
d(der(A), 0):-number(A).
d(der(X + Y), DX + DY):-d(der(X), DX), d(der(Y), DY).
d(der(X * Y), X * DY + Y * DX):-d(der(X), DX), d(der(Y), DY).
d(der(der(X)), DDX):-d(der(X), DX), d(der(DX), DDX).
```

2. Consider McCarthy's 91 function [Knu91] with the query `mc_carthy_91(Argument, Value)`, where the first argument is an integer.

```
mc_carthy_91(X, Y)  :-  X > 100, Y is X - 10.
mc_carthy_91(X, Y)  :-  X =< 100, Z1 is X + 11,
                        mc_carthy_91(Z1, Z2), mc_carthy_91(Z2, Y).
```

For programs, such as repetitive derivations, we can show termination by considering *recursively definable sizes,* i.e., the size of a term depends only on its functor and the sizes of its arguments. These recursively definable sizes, which are not necessarily numerical, are a proper extension of the sizes induced by linear norms. For instance, the depth of a term is a recursively definable size, but it cannot be expressed by any linear norm. Norms that are based on recursively definable sizes can be incorporated into the framework of *TermiLog* and similar systems [LSS97, CT97].

For programs, such as McCarthy's 91 function, termination depends on arithmetic calculations, and showing termination is difficult, since the usual order for the integers is not well-founded. For

*Tel-Aviv University, Tel-Aviv, Israel, e-mail:nachumd@math.tau.ac.il
†Hebrew University, Jerusalem, Israel, e-mail:{naomil,sagiv,alicser}@cs.huji.ac.il
‡Contact author

such programs, we have developed the following method for showing termination. First, we deduce automatically a finite abstract domain representing the integers. For example, the integer abstractions for the 91 function consists of the following intervals: $\{X \leq 89, 89 < X \leq 100, X > 100\}$. Using this abstraction, we can apply an abstract interpretation to infer a finite number of atoms abstracting answers of the program. These abstract atoms serve as a basis for extending the technique of the query-mapping pairs [LS97]. Specifically, for each query-mapping pair that is suspected of expressing a non-terminating behavior, we guess a bounded integer-valued termination function. If, while traversing the pair, the termination function decreases monotonically, termination is shown. In our approach, it is sufficient to guess a simple termination function for each suspicious query-mapping pair, and that gives our approach an edge over the classical approach of using a *single* termination function, which inevitably has to be more complicated and is harder to guess automatically. Thus, our approach of combining a finite abstraction of the integers with the technique of the query-mapping pairs is essentially capable of dividing a termination proof into several cases, such that a simple termination function suffices for each case, and the whole process of proving termination can be done automatically in the framework of *TermiLog* and similar systems [LSS97, CT97].

References

[Apt97] Krzysztof R. Apt. *From Logic Programming to Prolog.* Prentice-Hall International Series in Computer Science. Prentice Hall, 1997.

[CT97] Michael Codish and Cohavit Taboch. A semantic basis for termination analysis of logic programs and its realization using symbolic norm constraints. In Michael Hanus, Jan Heering, and Karl Meinke, editors, *Algebraic and Logic Programming, 6th International Joint Conference, ALP '97 - HOA '97*, pages 31–45. Springer Verlag, 1997. Lecture Notes in Computer Science, volume 1298.

[DSD94] Danny De Schreye and Stefaan Decorte. Termination of Logic Programs: The Never-Ending Story. *The Journal of Logic Programming*, 19/20:199–260, May/July 1994.

[Knu91] Donald E. Knuth. Textbook Examples of Recursion. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation. Papers in Honor of John McCarthy*, pages 207–229. Academic Press, Inc., 1991.

[LS97] Naomi Lindenstrauss and Yehoshua Sagiv. Automatic termination analysis of logic programs. In Lee Naish, editor, *Proceedings of the Fourteenth International Conference on Logic Programming*, pages 63–77. MIT Press, July 1997.

[LSS97] Naomi Lindenstrauss, Yehoshua Sagiv, and Alexander Serebrenik. *TermiLog*: A system for checking termination of queries to logic programs. In Orna Grumberg, editor, *Computer Aided Verification, 9th International Conference*, pages 63–77. Springer Verlag, June 1997. Lecture Notes in Computer Science, volume 1254.

[Plü90] Lutz Plümer. *Termination Proofs for Logic Programs.* Lecture Notes in Artificial Intelligence, volume 446. Springer Verlag, 1990.

[UvG88] Jeffrey D. Ullman and Allen van Gelder. Efficient Tests for Top-Down Termination of Logical Rules. *Journal of the Association for Computing Machinery*, 35(2):345–373, April 1988.