Termination Semantics of Logic Programs with Cut and Related Features

Jamie Andrews
Dept. of Computer Science
Univ. of Western Ontario
London, Ontario, Canada

There are various varieties of termination for logic programs. I am particularly interested in the termination of logic programs which use practical features such as the Prolog "cut". In order to prove termination of such programs, it is necessary first to define a semantics for them. In my talk, I will discuss the various varieties of termination and the semantics proposed for characterizing them, leading to semantics of Prolog with cut which facilitates proofs of termination properties.

Varieties of Termination. The first, and most obvious, dimension in which we can classify logic programming termination is the result which is given upon termination. Goals can either succeed (returning one or more substitutions), fail (returning nothing), or diverge. There is also clearly only one way for goals to fail: the search tree must be finite, and all the leaf nodes of that tree must be failure nodes. Beyond that, we can identify several different senses in which a goal can succeed.

A goal can *existentially terminate*, which means that either it fails or at least one leaf node in the search tree is a success node. Alternatively, it can *universally* or *strongly terminate*, which means that the search tree is finite (every leaf node necessarily being a success or failure node). Clearly, the notion of existential termination is useful only when discussing breadth-first search strategies, whereas if a goal strongly terminates then we know that Prolog (for instance) can find a solution.

However, in fact neither notion characterizes what we might see as the natural concept of termination in Prolog. Prolog uses a depth-first, left-to-right search strategy, and can return a solution iff it finds one before the first infinite branch in the tree during its traversal. The user can then request further solutions on backtracking, but the computation of the query has succeeded at least once. Hence an insistence on strong termination disallows goals which, on backtracking, return a finite number of solutions and then diverge, or return an infinite number of solutions. Strong termination is "strong" in the sense of insisting on strong conditions for termination.

Most work on termination of logic programs has concentrated on strong termination. However, since the depth-first, left-to-right search strategy is used by most logic programming systems, the problem of characterizing this strategy is also important.

The Prolog Cut. Characterizing depth-first, left-to-right termination is especially important in the context of programs which use the Prolog "cut" or related features. These features cut away part of the search tree. Which part is cut depends on what is the current node in the depth-first traversal.

The cut (!) has been a feature of Prolog implementations for over fifteen years. When used as a subgoal in a clause body, it has the effect of cutting away all alternative clauses to the current clause, and (in the case of "hard" cut) all backtrack points encountered since the processing of the current clause started. Most of the uses of cut in typical Prolog programs have the effect of if-then-else expressions; we can therefore replace clauses which use cut for this purpose with clauses which use negation as failure. Termination of goals with respect to these programs can then be characterized by some semantics of programs with negation as failure under depth-first termination, such as the author's [And97], or strong termination, such as Stärk's [Stä98].

However, cut is also used for other purposes in practical logic programs. It is used when the programmer knows that all solutions being returned by a previous subgoal will have the same effect as the first solution. It is also used for efficiency, to cut away any backtrack points left over from a previous subgoal, which might otherwise occupy memory. Finally, it is often used to select the first solution returned by the previous subgoals in the clause. It is for this latter use that it is important to characterize depth-first termination, since cut selects the first solution returned by this strategy.

Characterizing Cut. Given the apparently non-logical nature of the Prolog cut, it is interesting that we can characterize a large subset of the programs using it with a semantics similar to that used to logically characterize depth-first Prolog with negation as failure [And97]. We can adapt the preliminary technique reported on in [And95] as follows.

We begin with a generalization of simple Prolog programs with cut, and an operational semantics of this generalization. We then show that we can transform any such program into a "completed" form which has the same behaviour but in which the predicates have only one clause each; a new "if-then" construct is introduced for this purpose. We then place a mode restriction on the completed program similar to Stärk's for Prolog with negation as failure [Stä98]. The mode restriction disallows the most egregiously non-logical Prolog programs which use cut, but still allows the first-solution and efficiency behaviours which hard cut gives us.

We are then able to characterize the mode-restricted, completed programs with a style of semantics referred to as "unfolding-normal-form-valuation" (UNV) semantics in [And97]. Given a goal G and a program P, we consider all unfoldings G' of G with respect to P. We put these unfoldings in a normal form, and apply a valuation (compositional function from goals to truth values) to the normal forms. The program valuation v_P is then defined as the function which returns the "most true" of these truth values, where "true" is more true than "undefined", which is more true than "false". The UNV semantics characterizes the operational semantics in the sense that a goal G succeeds (fails, diverges) with respect to a program P in the operational semantics iff $v_P(G)$ is "true" ("false", "undefined"). Since the operational semantics uses the depth-first, left-to-right strategy, we can claim to have abstractly characterized depth-first termination with respect to this version of cut.

References

[And95] James H. Andrews. A paralogical semantics for the Prolog cut. In *Proceedings of the International Logic Programming Symposium*, Portland, December 1995. MIT Press.

[And97] James H. Andrews. A logical semantics for depth-first Prolog with ground negation. *Theoretical Computer Science*, 184(1-2):105–143, September 1997.

[Stä98] Robert Stärk. The theoretical foundations of LPTP (a logic program theorem prover). *Journal of Logic Programming*, 36(3):241–269, 1998.