3 E − 4

A Termination Function of Recursive Programs and Its Application to Partial Evaluation

Litong Song Yoshihiko Futamura

Department of Information and Computers, School of Science and Engineering, Waseda University

1. Introduction

We propose a termination function for terminating a class of recursive programs. When a given program is not in the class, we can define an effective termination condition for the program based on our function. The function is also applicable in terminating partial evaluation.

2. The Termination of Recursive Function

The termination of recursive function is an "ancient" and undecidable problem. According to Z.Manna's book [3], for any terminable recursive function, there exists a termination function s which satisfies relation s(X) > s(X'), where function is in the form of f(X)=⋯f(X')⋯. Practically, it is very difficult to find such a termination function as s. In partial evaluation the termination of recursive function specialization is also an important and difficult problem. For instance, a recursive function: f(m,n)= **if** mod(m,n)=0 **then** f(m/n, n+1)+1 **else** 1. If n=2 and m≧0, the function can be unfolded forever.

In order to solve the problem, there have been some better papers to discuss the problem such as Futamura's paper about GPC[2]. The paper discussed the essence of generalized partial evaluation (GPC) and at the same time presented a kind of general halting condition in GPC. Because the paper is aimed at GPC, it is inevitable to have some defects and insufficiency for the partial evaluation with given values. For instance, if n≧0, the unfolding of the function above can't stop forever. Many other interrelating papers, for example in [1], only paid attention to given values so that might result in inefficiency. The paper takes into account not only given values but also the logical structure of the functions to be partially evaluated.

3. A Termination Function of Recursive Function

Suppose recursive function is generally in the form as:

f(X,Y)= ⋯ **if** R(X) **then** ⋯ f(X',Y') ⋯ **else** ⋯, where X, Y, and R(X) (in the form of $e_1(X)$ R $e_2(X)$) be known, unknown argument and relation expression respectively.

Definition3.1: s: Union_Domain × Bool_EXP → INT, Union_Domain=INT+REAL+SET. s(X,R(X))=

if R="≠"

then if type(X)=SET **then** num($(e_1(X)-e_2(X)) \cup (e_2(X)-e_1(X))$)

else num{u| $(e_1(X) < u \leqq e_2(X)) \vee (e_1(X) \geqq u > e_2(X))$}

else num($\{u|\ e_1(X)\ R^L\ u\ R^R\ e_2(X)\} \cup \{u|\ e_2(X)\ R^L\ u\ R^R\ e_1(X)\}$)

Where num(Set)=the number of elements in Set and R^L, R^R are defined in appendix. Based on s, a well-founded ordering and corresponding unfolding condition can be defined.

Definition3.2: For any known arguments X, X' and a relation operator R, a well founded ordering $\angle_R$ is defined as:

$X' \angle_R X$ **iff** $s(X',R(X')) < s(X,R(X))$

Definition3.3(*Unfold_Cond*): For any recursive function definition in the form above, recursive call f(X',Y') can be further specialized or unfolded **iff** $X' \angle_R X$.

Theorem3.1: The existence of the well founded ordering $\angle_R$ is sufficient to ensure that the specialization of recursive function f terminates (provable with inductive method).

In addition to the condition, we also present an unfolding rule as follow to ensure unfolding termination and efficiency.

***Rule*:** This is just so called Termination-on-the-Second-Call principle. "If current recursive call is in GPC process of an equivalent function call, terminate unfolding".

4. The Expansion to Unfolding Condition

In section 3, R(X) is only a relation expression but not a complete Boolean expression. The Boolean expression can generally be indicated with following normal form:

B::=and and::=or | and∧and or::=R | or∨or

If $B(X)=or(X)=R_1(X) \vee \cdots \vee R_n(X)$, then $\exists_{1 \leq i \leq n}(R_i(X) \rightarrow R_1(X) \vee \cdots \vee R_n(X))$. Therefore, the following definition about relation $\angle_B$ can be inferred.

Definition4.1: For a $R_1(X) \vee \cdots \vee R_n(X)$. $X' \angle_B X$ **iff**

$s((X),R_1(X) \vee \cdots \vee R_n(X)) < s(X',R_1(X') \vee \cdots \vee R_n(X'))$

Where $s((X),R_1(X)\vee\cdots\vee R_n(X))=\sum_{i=0}^{n}(R_i(X)\rightarrow s(X,R_i(X)),0)$

If $B(X^m)=and(X^m)=or_1(X^m)\wedge\cdots\wedge or_n(X^m)$ (X^m indicates the value of X after function f has just been unfolded for m times), then $\exists_{1\le i\le n}(\neg or_i(X^m)\rightarrow\neg or_1(X^m)\wedge\cdots\wedge or_n(X^m))$. Therefore $\exists_{1\le i\le n}\forall_{1\le j\le m}(s(X^j,or_i(X^j))>s(X^{j'},or_i(X^{j'})))$ should be an unfolding condition. In fact, it isn't certain to exist such an i($1\le i\le n$) that $\forall_{1\le j\le m}s(X^j,Or_i(X^j))$ always decreases monotonously, so the condition easily leads to insufficient specialization. In many cases, $s(X,or_i(X))$ sometime increases and sometime decreases, but $\sum_{i=0}^{n}s(X,or_i(X))$ always decreases and finite, so we can infer a more effective definition for $\angle_B$.

Definition4.2: For a relation $or_1(X^m)\wedge\cdots\wedge or_n(X^m)$. $X'\angle_B X$ iff

$$\left(\sum_{i=0}^{n}s(X^{m'},or_i(X^{m'}))<\sum_{i=0}^{n}s(X^m,or_i(X^m))\right)$$
$$\vee\exists_{1\le i\le n}\forall_{1\le j\le m}(s(X^{j'},or_i(X^{j'}))<s(X^j,or_i(X^j)))$$

5. Conclusion

Because the method presented in the paper pays more attention to the given values and logical structure of recursive function, it shows more powerful function than [1] in partial evaluation with given values. Of course, there still exist some insufficient aspects in our method. In the sense of $\angle_B$, the variation of X must be monotonously decreased. This restriction may lead to unfolding termination too early and so the residual function generated may be not the best one. For example, for a function definition: $f(x_1,x_2,y)$=**if** $x_1>0$ **then** y **else** $y\times f(x_1+x_2,x_2+1,y-1)$ and a function call f(-2,-1,y). Its unfolded result is: $f(-2,-1,y)=y\times f(-3,0,y-1)$. In fact, the function can be further unfolded into: $y\times(y-1)\times(y-2)\times\cdots\times(y-5)$. In practical application, there must not be the confine that X' and X must keep monotonous relation in the sense of $\angle_B$. We can add a new unfolding condition to ***Unfold_Cond***.

Definition5.1: For any known parameters X, X' and a relation B. $X'\angle^*_B X$ iff $s((X)',B((X')'))-s(X',B(X'))<s(X',B(X'))-s(X,B(X))$

Definition5.2(*Unfold_Cond+*): recursive call f(X',Y') can be further specialized or unfolded iff $X'\angle_B X$ or $X'\angle^*_B X$.

By the new definition, although $X'\angle_B X$ possibly doesn't hold, if it is true for $s((X')',B((X')'))-s(X',B(X'))<s(X',B(X'))-s(X,B(X))$, it will mean that in the sense of $\angle_B$ the augmentation of X gradually decreases and due to the finitary characteristics of s(X',B(X'))-s(X,B(X)), after unfolding function for numerous times, X maybe tend to decrease finally and contrarily begin to satisfy ***Unfold_Cond***. By ***Unfold_Cond+***, the example above can be specialized to end. Let us look at a figure.

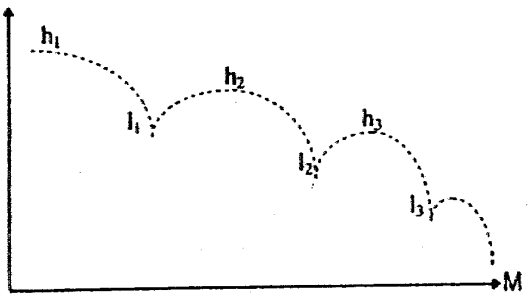

Figure1

This is a curve figure of s(X,B(X)) with respect to M(unfolding times). By Definition3.1 and Definition3.2, we can only unfold function f(X,Y) between point h_1 and l_1. With Definition5.1 and Definition5.2, the problem can be solved. Nevertheless, it is necessary to add some restrictions to the definition above. For instance, s(X,B(X)) at point h_1 is greater than that at h_2, s(X,B(X)) at l_1 greater than that at l_2, and so on.

Under the premise of halting, the weaker unfolding condition is, the wider usage scope is, and the more efficient residual function is. Our future work is just to further expand these unfolding conditions to some suitable extent and so as to make our method more practical.

Reference

[1]. "Partial Evaluation", International Seminar, Dagstuhl Castle, Germany, February 1996 Selected Papers,Springer. .

[2]. Futamura,Y.,K.Nogi and A.Takano, Essence of generalized partial computation. *Theoretical Computer Science*(1991)61-79

[3]. Z.Manna, The Logic of Computer Programming, SIAM, phil. PA., 1980.

Appendix

$<^L=<$, $\le^L=\le$, $=^L=\le$, $\ne^L=<$, $\subset^L=\subset$, $\subseteq^L=\subseteq$,

$<^R=\le$, $\le^R=\le$, $=^R=\le$, $\ne^R=\le$, $\subset^R=\subseteq$, $\subseteq^R=\subseteq$