

# TERMINATION OF QUERIES

## 1. Introduction

more material in:

D. De Schreye, S. Decorte, Termination of Logic Programs: the Never Ending Story, *J. Logic Programming*, **19-20**, 1994, 199–260.

K.R. Apt and D. Pedreschi. Reasoning about Termination of Pure Prolog Programs. *Information and Computation*, **106**:109–157, 1993.

K.R. Apt, M. Bezem Acyclic Programs. *New Generation Computing* **9**, 335–363 (1991)

M. Bezem. Strong termination of logic programs. *Journal of Logic Programming*, **15**:79–97, 1993.

Byron Cook, Andreas Podelski, Andrey Rybalchenko. Proving program termination. *CACM*, **54**(5):88-98. 2011

motivation

- completeness results are weak, they do not guarantee anything about termination
- The Kowalski slogan:  
ALGORITHM = LOGIC + CONTROL  
logic: the declarative semantics  
control: efficient “proofs”, termination is a minimum

caveat: in general undecidable

simple and well-known incompleteness result

Assume there exists a function  
*terminates*: Program  $\rightarrow$  Bool  
which always terminates and returns *true* when  
its input is a terminating program

program *P*

L: **if** *terminates*(*P*) **then go to** L  
**else stop**

the assumption leads to a contradiction

If *terminates* returns true then *P* loops for ever

if *terminates* returns false, then *P* terminates

## Two research lines

- Necessary and Sufficient conditions  
so conditions must be undecidable; manual methods  
(in Sections 2 and 3)
- Decidable sufficient conditions  
can be automated; can be integrated in other tools; e.g.  
program transformation should preserve declarative seman-  
tics (partial correctness) but also termination (total cor-  
rectness)  
( in Section 4)

termination does depend on the query:  
a program can terminate for one query, can  
loop for others

several kinds of termination

- universal termination  
finds all solutions and terminates
- existential termination  
either finite failure  
or at least one solution  
after first solution, program may loop  
universal termination implies existential termination  
proving existential termination (in absence of universal ter-  
mination) is much harder because order of branches in  
SLDNF trees must be taken into account
- strong termination  
program terminates independent of com-  
putation rule  
most simple, but does not hold so often

$append([], X, X) \leftarrow$   
 $append([X|U], V, [X|W]) \leftarrow append(U, V, W)$

universal termination for

$\leftarrow append([1, 2], [3], L)$     1 solution  
 $\leftarrow append([1, 2], [3], [1, 2, 3])$     1 solution  
 $\leftarrow append([1, 2], [3], [1, 3])$     no solution  
 $\leftarrow append(L_1, L_2, [1, 2, 3])$     many solutions

existential termination for

(order of clauses/branches important)

$\leftarrow append(L_1, [1, 2, 3], L_3)$      $\infty$  solutions  
 $\leftarrow append([X|Y], Y, [Z|Y])$     1 solution ( $Y=[]$ )

universal termination under PROLOG computation rule for

$\leftarrow append([1, 2], [3], L), append(L, [4, 5], LL)$

no termination for

$\leftarrow append([X|Y], [], Y)$     is nonlinear in  $Y$

basic idea of any termination proof

independent of language

## *well-founded ordering*

well-founded set – well-founded order

- partial order
- decreasing sequences are finite

take as elements the states of a computation

as order the execution order

if well-founded then it terminates

how to prove?

map states of computation to a set which is known to be well-founded and show:

$\forall$  states  $s_1, s_2$  if  $s_2$  is successor of  $s_1$  then  $f(s_2) < f(s_1)$

if so, then

the set of states of the computation (the trace) is a well-founded set

the computation terminates

allowed to remove finite sequences from the trace

Lemma

every finite computation can be mapped on a decreasing sequence in a well-founded ordering

Proof: is a finite sequence  $s_0, s_1, \dots, s_n$

$$f(s_i) = n - i$$

we have  $f(s_i) = n - i > f(s_{i+1}) = n - i - 1$   
and  $\mathcal{N}$  is well-founded

need proof without building trace (may not terminate)

Example: For programs without variables:

define size: ground atom  $\rightarrow \mathcal{N}$

if for all clauses  $A \leftarrow B_1, \dots, B_n$

$$\forall i : size(A) > size(B_i)$$

then well-founded order over the nodes of any SLD-tree

Proof:

- define  $f(\leftarrow A_1, \dots, A_n) = \{size(A_1), \dots, size(A_n)\}$   
(not set but *multiset*: repetition of elements!)  
order relation:  $\{a\} \cup S > \{b_1, \dots, b_n\} \cup S$  if  $\forall i : a > b_i$   
multiset is well founded
- resolvent of  $\leftarrow A_1, \dots, A_i, \dots, A_n$  is  $\leftarrow A_1, \dots, B_1, \dots, B_n, \dots, A_n$
- we have  
 $f(\leftarrow A_1, \dots, A_i, \dots, A_n) > f(\leftarrow A_1, \dots, B_1, \dots, B_n, \dots, A_n)$   
thus the SLD-tree is finite

Also for programs with variables?

Let  $size(p(t_1, \dots, t_n)) =$  number of functors

for all ground substitutions  $\theta$ :

$$size(append([X|U], V, [X|W])\theta) > \\ size(append(U, V, W)\theta)$$

but non-terminating queries!

e.g.  $\leftarrow append(L_1, [], L_3)$

one resolution step yields the goal

$\leftarrow append(U, [], W)$

but instantiates the original goal to

$\leftarrow append([X|U], [], [X|W])$

the “backpropagation” effect of unification

unbounded increase in size of original goal

does not happen when unification is one way:

instantiating variables in head but not in original goal

→ let “size” look only at (parts of) terms of calls which are not further instantiated

Another difficulty:  
variables local to body

$$\text{perm}([X|Y], [U|V]) \leftarrow \text{del}(U, [X|Y], W) \\ \text{perm}(W, V)$$

calls to *perm/2* with first argument ground:  
size cannot consider second argument  
due to backpropagation

first argument: no size possible such that  
 $\text{size}([X|Y]) > \text{size}(W)$   
for all ground substitutions

termination only when leftmost atom selected  
→ will need to find relationship  
created by call  $\text{del}(U, [X|Y], W)$   
between  $\text{size}([X|Y])$  and  $\text{size}(W)$

## 2. Termination for DEFINITE programs

### 2.1 Strong Termination

Definition: Level mapping

a function  $f : B_P \rightarrow \mathcal{N}$

maps every ground atom to *finite* value!

Definition: Recurrent program

$P$  is recurrent if

there exists a level mapping  $f$   
such that for every

$A \leftarrow B_1, \dots, B_n \in \text{ground}(P)$ :  
 $f(A) > f(B_i)$  for each  $i$

Theorem:

$P$  is recurrent **iff**

$P$  is terminating for all ground queries  $\leftarrow A$

is independent of computation rule!

strong termination

Proof

- ASSUME  $P$  terminating for all ground queries  $\leftarrow A$ 
  - let  $f(A) =$  length longest derivation in SLD-tree of  $\leftarrow A$
  - for each ground rule  $A \leftarrow B_1, \dots, B_n$ :  
longest derivation for  $\leftarrow A >$  longest derivation for  $\leftarrow B_i$   
i.e.  $f(A) > f(B_i)$ , i.e.  $P$  is recurrent

proof cntd

- ASSUME  $P$  is recurrent (hence level mapping  $f$  exists)
  - assume an infinite derivation  $\leftarrow A, \leftarrow G_1, \leftarrow G_2, \dots$  exists
  - apply a grounding substitution  $\theta$  such that the derivation becomes ground
  - let  $\leftarrow G_i\theta = \leftarrow A_1\theta, \dots, A_i\theta, \dots, A_m\theta$   
let  $\leftarrow G_{i+1}\theta = \leftarrow A_1\theta, \dots, B_1\theta, \dots, B_k\theta, \dots, A_m\theta$   
recurrent: for all  $j$ :  $f(A_i\theta) > f(B_j\theta)$
  - let  $size(\leftarrow A_1\theta, \dots, A_i\theta, \dots, A_m\theta) = \{f(A_1\theta), \dots, f(A_i\theta), \dots, f(A_m\theta)\}$  (multiset)
  - $size(\leftarrow G_i\theta) > size(\leftarrow G_{i+1}\theta)$  according to order relation over multisets, thus the successive states in the derivation are a well-founded set and the derivation is finite
  - contradicts with assumption of infinite derivation

Example:

$append([X|U], V, [X|W]) \leftarrow append(U, V, W)$

level mapping:  $||$ :

$|append(t_1, t_2, t_3)| = length(t_1)$  where

$length(t) = \mathbf{if } t = [t_1|t_2] \mathbf{ then } 1 + length(t_2)$   
 $\mathbf{else } 0$

$length([X|U]\theta) = 1 + length(U\theta) > length(U\theta)$

thus  $append$  is recurrent

so we can prove strong termination of all ground queries

what about non-ground queries?

Definition: Boundedness

let  $f$  be a level mapping

an atom  $A$  is *bounded* wrt  $f$  if

$\{f(A\theta) \mid A\theta \text{ is ground}\}$  is bounded in  $\mathcal{N}$

has a finite upper bound

boundedness means back propagation cannot infinitely increase

the size of the original query

all ground atoms are bounded!

Theorem

if  $P$  is recurrent wrt level mapping  $f$  then  $P$  is strongly terminating for all atomic queries which are bounded wrt  $f$

Proof:

- assume a bounded query for which an infinite derivation exist
- apply a grounding substitution on it
- contradiction with recurrency of  $P$

## Theorem

if atomic queries bounded wrt  $f$  always strongly terminate then  $P$  is recurrent wrt some level mapping  $g$

Proof

- All ground queries are bounded wrt  $f$  hence strongly terminate
- Hence  $P$  is recurrent wrt some level mapping (according to a previous theorem)

There are many level mappings for which `append` is recurrent:

With  $append(t_1, t_2, t_3)$  a ground atom,

$$|append(t_1, t_2, t_3)| = length(t_1),$$

$$|append(t_1, t_2, t_3)| = length(t_3),$$

$$|append(t_1, t_2, t_3)| = size(t_1),$$

$$|append(t_1, t_2, t_3)| = size(t_3),$$

$$|append(t_1, t_2, t_3)| = \min(size(t_1), size(t_3)),$$

$$|append(t_1, t_2, t_3)| = \min(length(t_1), length(t_3)).$$

The last is the most useful because any query bounded by one of the others is also bounded by the last one

## Exercise

1. Consider the following program:

$\text{update}(K, V, [ ], [p(K, V)]) \leftarrow.$

$\text{update}(K, V, [p(K, V1)|L], [p(K, V)|L]) \leftarrow.$

$\text{update}(K, V, [p(K1, V1)|L], [p(K1, V1)|L1]) \leftarrow K \neq K1, \text{update}(K, V, L, L1).$

Find a level mapping for which the program is recurrent and from which termination follows for the queries

$\leftarrow \text{update}(3, [a, b, c], [p(1, a), p(2, f(b))], L).$  and

$\leftarrow \text{update}(3, c, L, [p(1, a), p(2, b), p(3, c), p(4, d)]).$

## 2.2 LD-termination

Strong termination (independent of computation rule) is rather exceptional

i.e. clauses with local variables:

$$p(X) \leftarrow \dots, p(Y), \dots$$

no level mapping exists such that

$$|p(X)\theta| > |p(Y)\theta| \text{ for every grounding } \theta$$

need to fix computation rule

we take the PROLOG one:

always select leftmost atom: *LD-derivations*

*left termination*: all LD-derivations terminate

“good” programs are left terminating

$$\begin{aligned} perm([X|Y], [U|V]) \leftarrow & del(U, [X|Y], W) \\ & perm(W, V) \end{aligned}$$

$\leftarrow perm(L, P)$  is left terminating when  $L$  is bounded by *length*

to prove it, we need to study by what  $W$  can be instantiated after  $del(U, [X|Y], W)$  is solved

Definition: acceptability

$P$  is acceptable if there exists a level mapping  $f$  and a model  $I$  such that for each clause  $A \leftarrow B_1, \dots, B_k \in \text{ground}(P)$ :

**if**  $I \models B_1, \dots, B_{i-1}$  **then**  $f(A) > f(B_i)$  for all  $i$

Theorem

$P$  is acceptable **iff**  $P$  is left terminating for all ground queries  $\leftarrow A$

Proof

- ASSUME  $\leftarrow A$  left terminating, prove acceptable
  - let  $f(A)$  be the length of the longest derivation (under the computation rule) of  $\leftarrow A$
  - consider ground clause  $A \leftarrow B_1, \dots, B_k$ :  
if  $B_1, \dots, B_{i-1}$  succeed, i.e. true in  $M_P$ , a model of  $P$  then  $f(B_i) < f(A)$
  - thus  $P$  acceptable wrt level mapping  $f$
- ASSUME acceptable program (i.e. level mapping exists) , prove left terminating
  - assume infinite derivation  $\leftarrow A, \leftarrow B_1, \dots, B_k, \leftarrow \dots$  exists
  - apply grounding  $\theta$  on it
  - there is a ground clause  $A \leftarrow B_1\theta, \dots, B_k\theta$
  - infinite derivation for  $\leftarrow B_1\theta, \dots, B_k\theta$ , so there exists a  $i$  such that  $\leftarrow B_1\theta, \leftarrow B_{i-1}\theta$  succeeds and  $\leftarrow B_i\theta$  does not terminate
  - i.e.  $M_P \models B_1\theta, \dots, B_{i-1}\theta$  and  $M_P \subseteq I$  thus due to acceptability:  $f(A) > f(B_i\theta)$
  - By induction, one can construct an infinite decreasing sequence  $f(A) > f(B_i\theta) > f(C_j\sigma) \dots$  of natural numbers which is a contradiction

## Theorem

if  $P$  is acceptable wrt level mapping  $f$  and model  $I$  then  $P$  is left terminating for all atomic queries which are bounded wrt  $f$

Proof: an infinite derivation for a bounded query can be instantiated into an infinite derivation for a ground query, contradicts acceptability

## Theorem

if atomic queries bounded wrt  $f$  always left terminate then  $P$  is acceptable wrt some level mapping  $g$

Proof

- all ground queries are bounded wrt  $f$  hence left terminate
- thus  $P$  is acceptable (according to a previous theorem)

complete but –undecidability of termination–  
no “algorithm” is possible for constructing the level mappings

modular proofs are possible

## Example: a permutation procedure

(1)  $p([], []) \leftarrow$

(2)  $p(Xs, [X|Ys]) \leftarrow a(X1s, [X|X2s], Xs), a(X1s, X2s, Zs), p(Zs, Ys).$

(3)  $a([], Ys, Ys) \leftarrow$

(4)  $a([X|Xs], Ys, [X|Zs]) \leftarrow a(Xs, Ys, Zs).$

- the program is not recurrent

$\leftarrow p([a, b], p[b, a])$  (a ground query)

$\leftarrow a(X1s, [b|X2s], [a, b]), a(X1s, X2s, Zs), p(Zs, [a])$

solving first  $a(X1s, X2s, Zs)$  leads to infinite branch

- the program is acceptable

– a level mapping  $||$ :

$$|p(t1, t2)| = length(t1) + 1$$

$$|a(t1, t2, t3)| = \min(length(t1), length(t3))$$

– a Herbrand interpretation (a subset of  $B_P$ )

$$I = \{p(s, t) \mid s, t \in U_L\} \cup$$

$$\{a(t1, t2, t3) \mid t1, t2, t3 \in U_L,$$

$$length(t1) + length(t2) = length(t3)\}$$

## Proof of acceptability:

- $I$  is a model it is larger than  $M_P$ , one does not need the minimal model!
  - $p/2$  clauses: trivial because head is true for every variable assignment
  - (3): for every variable assignment  $\theta$ :  
 $0 + \text{length}(Ys\theta) = \text{length}(Ys\theta)$  hence  $a([], Ys\theta, Ys\theta) \in I$
  - (4): for every variable assignment  $\theta$ :  
If  $a(Xs\theta, Ys\theta, Zs\theta) \notin I$  then clause is trivially true  
If  $a(Xs\theta, Ys\theta, Zs\theta) \in I$  then  
 $\text{length}(Xs\theta) + \text{length}(Ys\theta) = \text{length}(Zs\theta)$   
and thus  
 $1 + \text{length}(Xs\theta) + \text{length}(Ys\theta) =$   
 $1 + \text{length}(Zs\theta)$  hence  
 $a([X\theta|Xs\theta], Ys\theta, [X\theta|Zs\theta]) \in I$  and the clause is true.
- $a/3$  is recurrent wrt  $\parallel$   
(4): for every variable assignment  $\theta$ :  
 $\min(1 + \text{length}(Xs\theta), 1 + \text{length}(Zs\theta)) >$   
 $\min(\text{length}(Xs\theta), \text{length}(Zs\theta))$

- $p/2$  is acceptable wrt  $\|\cdot\|$  and  $I$

Let  $\theta$  be a grounding substitution for (2)

- $|p(X_s, [X|Y_s])\theta| \stackrel{?}{>} |a(X1_s, [X|X2_s], X_s)\theta|$   
 $length(X_s\theta) + 1 \stackrel{?}{>} \min(length(X1_s\theta), length(X_s\theta))$   
*true* because  $length(X_s\theta) + 1 > length(X_s\theta)$
- assume  $I \models a(X1_s, [X|X2_s], X_s)\theta$   
 $|p(X_s, [X|Y_s])\theta| \stackrel{?}{>} |a(X1_s, X2_s, Z_s)\theta|$   
 $length(X_s\theta) + 1 \stackrel{?}{>} \min(length(X1_s\theta), length(Z_s\theta))$   
 from assumption:  $length(X1_s\theta) + 1 + length(X2_s\theta) = length(X_s\theta)$   
 thus *true* because  $length(X_s\theta) + 1 > length(X1_s\theta)$
- assume  $I \models a(X1_s, [X|X2_s], X_s)\theta, a(X1_s, X2_s, Z_s)\theta$   
 $|p(X_s, [X|Y_s])\theta| \stackrel{?}{>} |p(Z_s, Y_s)\theta|$   
 $length(X_s\theta) + 1 \stackrel{?}{>} length(Z_s\theta) + 1$   
 from assumption:  $length(X1_s\theta) + 1 + length(X2_s\theta) = length(X_s\theta)$   
 and  $length(X1_s\theta) + length(X2_s\theta) = length(Z_s\theta)$   
 thus *true* because  $length(X_s\theta) > length(Z_s\theta)$

a bit tricky to find the right level mappings

## Exercise

2. Consider the following program:

$qs([], []) \leftarrow$ .

$qs([X|L], S) \leftarrow \text{part}(X, L, L1, L2),$   
 $\quad \text{qs}(L1, S1), \text{qs}(L2, S2),$   
 $\quad \text{append}(S1, [X|S2], S).$

$\text{part}(X, [], [], []) \leftarrow$ .

$\text{part}(X, [Y|L], [Y|L1], L2) \leftarrow X \geq Y, \text{part}(X, L, L1, L2).$

$\text{part}(X, [Y|L], L1, [Y|L2]) \leftarrow X < Y, \text{part}(X, L, L1, L2).$

$\text{append}([], L, L) \leftarrow$ .

$\text{append}([X|U], V, [X|W]) \leftarrow \text{append}(U, V, W).$

Show that the program is left terminating for all queries  $qs(t1, t2)$  where  $t1$  is a list with a bounded length.