UNIVERSITY OF OSLO
Department of informatics

Criticality Conditions on Equations to Ensure Poly-time Functions

Vuokko-Helena Caseiro

Research report 225

ISBN 82-7368-139-4
ISSN 0806-3036

November 1996

Criticality Conditions on Equations to Ensure Poly-time Functions[*]

Vuokko-Helena Caseiro
University of Oslo, Department of informatics
P. O. Box 1080 Blindern, N-0316 Oslo, Norway
Tel: +4722852405 Fax: +4722852401 E-mail: vuokko@ifi.uio.no

November 1996

Contents

[*]This work was partially supported by The Research Council of Norway.

1 Introduction

We consider equations defining functions on data structures built from constructors, e.g. sorting lists constructed from nullary nil and binary cons

$$\begin{array}{lcl}
\mathsf{quicksort}\,\mathsf{nil} & = & \mathsf{nil}\\
\mathsf{quicksort}\,(\mathsf{cons}\,x\,y) & = & \mathsf{qs}\,y\,x\,\mathsf{nil}\,\mathsf{nil}\\
\mathsf{qs}\,\mathsf{nil}\,z\,l\,r & = & \mathsf{append}\,(\mathsf{quicksort}\,l)\,(\mathsf{cons}\,z\,(\mathsf{quicksort}\,r))\\
\mathsf{qs}\,(\mathsf{cons}\,x\,y)\,z\,l\,r & = & \mathsf{if\text{-}then\text{-}else}\,(\mathsf{less}\,x\,z)\,(\mathsf{qs}\,y\,z\,(\mathsf{cons}\,x\,l)\,r))\\
 & & (\mathsf{qs}\,y\,z\,l\,(\mathsf{cons}\,x\,r))\\
\mathsf{append}\,\mathsf{nil}\,z & = & z\\
\mathsf{append}\,(\mathsf{cons}\,x\,y)\,z & = & \mathsf{cons}\,x\,(\mathsf{append}\,y\,z)\\
\mathsf{if\text{-}then\text{-}else}\,\mathsf{true}\,x\,y & = & x\\
\mathsf{if\text{-}then\text{-}else}\,\mathsf{false}\,x\,y & = & y
\end{array}$$

or exponentiation on unary numbers built from nullary 0 and unary succ

$$\begin{array}{lcllcl}
\mathsf{exp}\,(\mathsf{succ}\,x) & = & \mathsf{double}\ (\mathsf{exp}\,x) & \mathsf{exp}\,\mathsf{0} & = & \mathsf{succ}\,\mathsf{0}\\
\mathsf{double}\,(\mathsf{succ}\,x) & = & \mathsf{succ}\,(\mathsf{succ}\,(\mathsf{double}\ x)) & \mathsf{double}\ \mathsf{0} & = & \mathsf{0}
\end{array}$$

We know that sorting may be executed in polynomial time (poly-time), whereas exponentiation cannot. Is there some way of detecting this by a syntactical consideration of the equations?

In [2] we defined a class *poly-basic* of functions (on any data structure) characterized by having polynomial bounds i) on the number of "needed" calls to mutually recursive functions and ii) on the length of "needed" intermediate results. We showed that all functions in *poly-basic* are poly-time. But *poly-basic* is not a syntactically defined class. In examples it's often easy to see that (i) holds, but checking that (ii) holds might be just as difficult as checking poly-time. In this report we wish to develop some methods for checking (ii).

We will take an important idea from work by Bellantoni and Cook [1], and Leivant [3]. They have given syntactic characterizations of classes of subrecursive functions by equational definitions; in particular, in [1] the data structure is the binary numbers and the class is exactly the poly-time functions, in [3] results are for arbitrary data structures but the classes become larger than poly-time[1]. In their work, the key idea is to control recursion by requiring that what we do recursion on (e.g. append's first argument), is of a different nature than the result of recursive calls (e.g. $(\mathsf{append}\, y\, z)$). They accomplish this by, in our words, marking the argument positions where the result of recursive calls is received, as *critical* and disallowing recursion on critical.

Our definition of the critical positions of a function f will be that these are the argument positions of f that (directly or indirectly) in some right-hand side (rhs) are filled with (the result of) a recursive call for some function. We give a simple "marking algorithm" for assigning marks, "critical" or "noncritical", to all positions of all functions of an equation set. A main point in our approach is that the equations are allowed to have a general shape.

This report starts by recalling the system *poly-basic* and continues by modifying *poly-basic* gradually towards a poly-time equational system *DDC*.

First we define formally the concept of "critical position". Then we use it to replace (ii) by the finer requirements that output lengths be polynomial in "needed" noncritical input and strictly linear in "needed" critical input (i.e. for every function f there's a polynomial function P such that for all ground terms $X_1, \ldots, X_n$: $|(f\, X_1 \ldots X_n)| \leq P(\sum_{i \text{ noncrit \& needed}} |X_i|) + \sum_{i \text{ crit \& needed}} |X_i|)$.

Then we study how to control the linearity of arguments in critical positions. We must confront the problem of doubling an argument. Basically, there are two ways for a function to double an argument. The first is by *recurring* on it. The idea of [1] and [3] was to forbid recursion on critical altogether, and so will we. The second way of doubling is by having a rhs *nonlinear in the variable* from the related left-hand side position, and then in some way or other using constructors of arity at least two to put the variable copies together. This problem was not considered by [1] and [3]. We propose a way of describing what is combined by constructors, and, in particular, whether two copies of the same variable are combined.

In this way we obtain the system *DDC* – "Don't Double Criticals" which we consider the main achievement of this report. All function in (pure) *DDC*

[1]In [3], the length of a constructor term t is taken as the height of t as a tree.

systems are *poly-basic*, therefore poly-time. *DDC* includes the Bellantoni-Cook functions (proved in Appendix 1), so also in *DDC* any function on binary numbers is definable. But *DDC* is more general, allowing equations of a general shape on arbitrary data structures, so that e.g. slightly modified versions of sorting functions like quicksort become definable.

2 Preliminaries: Terms, Equations, Functions

Given three disjoint sets, of variables, of constructors with arity and of functions with arity greater than zero, respectively, we define *terms* in the usual way: A variable is a term, and if $t_1, \ldots, t_n$ are terms and h is a constructor or a function, then $(h\, t_1 \ldots t_n)$ is a term. Furthermore $(h\, t_1 \ldots t_n)$ is called an *application* with h as *head* and the t_i's as *arguments* of h. s is a *subterm* of t if s is t, or if t is an application $(h\, t_1 \ldots t_m)$ and s is a subterm of some t_i. A *constructor term* is a term built only from constructors. A *ground term* is a term built only from constructors and functions.

Define a *canonical equation system* to be a set of equations such that each function f is defined by

$$(f\ (c\, y_1 \ldots y_m)\, x_2 \ldots x_n) = r$$

where $n \geq 1, m \geq 0$ and there's one equation for each constructor c, where $y_1, \ldots y_m, x_2, \ldots, x_n$ all are different variables and r is a term with variables among $y_1, \ldots y_m, x_2, \ldots, x_n$. We consider only finite systems. All our equations will be in this form. As shorthand notation, sometimes we instead define a function f by composition, $(f\, x_1 \ldots x_n) = t$, where $x_1, \ldots, x_n$ are different variables and t is a term with variables among $x_1, \ldots, x_n$. In examples we will permit defining functions just for some constructors (e.g. append only for first argument a list, and not e.g. a boolean), formally, the rhs of the remaining equations can be taken as e.g. the constructor false. For the rest of this section we assume that a canonical (equation) system is given.

When a function f occurs in the lhs of an equation then *the equation is for f*. If a function g occurs in the right-hand side (rhs) of an equation for f then f *calls* g. If there is a sequence $f_1, f_2, \ldots, f_n$ $(n \geq 1)$ of different functions such that f_1 calls f_2, $\ldots$,f_n calls f_1 then each f_i is *recursive*, and every two functions from the sequence are *mutually recursive*.

We set up a directed graph, *the dependency graph* where the nodes are the functions $\{f, g, \ldots\}$, and there's an arc $f \to g$ iff f calls g. Define the binary relation $\overset{+}{\to}$ to be the transitive closure of $\to$. Define $f \equiv g$ iff $f \overset{+}{\to} g$ and $g \overset{+}{\to} f$. Note that $f \overset{+}{\to} f$ iff f is recursive, and $f \equiv g$ iff f and g are mutually recursive.

Given a node f, define the *M-set*[2] for f to be $M_f = \{g \mid f \equiv g\} \cup \{f\}$. Define a binary relation $\triangleright$ (written infix) on M-sets S, T by $(S \triangleright T)$ iff $S \neq T$ and there is a node (function) s in S and a node (function) t in T such that $s \rightarrow t$. So $\triangleright$ is antisymmetric and by definition it is irreflexive. We will do induction using $\triangleright$, both upwards and downwards. *S is over T/T is below S* if $S \triangleright T$).

In an equation $e : l = r$ for a function f, if in r there is a subterm t such that t is $(g\,t_1 \ldots t_n)$ and $g \in M_f$, then t is a *recursive call term in e*.

The system is *terminating* if when we turn the equations into rewrite rules by orienting them from lhs to rhs, then for any term t, any rewriting sequence of t is finite. Obviously, the system has unique normal forms, and if the system is terminating then the normal form of a term t is denoted $t!$.

3 The System *poly-basic*

In 3.1 and 3.2 we refer some definitions and results from [2]. In 3.3 we explain how we want to go on developing *poly-basic*.

3.1 Needed Parts: Fit Units and Fit Right-hand Side Trees

Assume that the programmer has defined an n-ary function f. Then we ask the programmer additionally to suggest some *f-units*. An f-unit is a subset of $\{1, \ldots, n\}$ - with the intuition that each f-unit should indicate an argument combination of f that may be needed to compute one call $(f\,\overline{X})$ for ground terms $\overline{X}$. E.g. for if-then-else (see Introduction) the programmer might suggest the units $\{1, 2\}$ and $\{1, 3\}$ since only arguments 1 and 2 *or* 1 and 3 will ever be needed to compute if-then-else. Formally $\{1, 2\}$ and $\{1, 3\}$ is an acceptable collection of if-then-else-units since each unit contains the position 1, and since each of the two rhs for if-then-else is uniquely "covered up" by one if-then-else-unit (x is covered by 2, y is covered by 3). Such formally acceptable units will be called *fit units*. Alternatively, $\{1, 2, 3\}$ alone would be ok, but less useful since it doesn't narrow the set of arguments.

Indeed, given a set of units they may be used to delimit which parts of a rhs might be needed: We draw each rhs as a tree in the usual way but in each function node choosing a unit U (among the given ones) and only including the children corresponding to U. Lemma 1 below states that when the given units are fit units, then for each function call $(f\,\overline{X})$ there is a unique such tree τ such that only things occurring in the "instantiated" τ are needed in order to compute $(f\,\overline{X})$.

[2] "M-set" means "set of mutually recursive functions"

Note 1 Many things in this report are "modulo what is needed" (argument positions, rhs subterms, trees of recursive calls ...). If things seem complicated, try to think of the special case when we call a function f on ground input $X_1, \ldots, X_n$ and *everything* is needed. Then a fit f-unit is just the trivial unit $\{1, \ldots, n\}$, a fit tree is just a subterm of the rhs (thought of as a tree), a fit based tree of recursive calls is the tree of *all* recursive calls on the initial call $(f\,\overline{X})$ - e.g. on $(\mathsf{quicksort}\,\mathsf{cons}\,6\,(\mathsf{cons}\,2)\,\mathsf{nil})$, the tree has root $\mathsf{quicksort}$ who has one child qs who has two children qs and so on (altogether 14 nodes).

In the following definitions we make precise the concepts of need.

3.1.1 Definitions.

Let a canonical system with equation set E be given. For an n-ary function f, define an *f-unit* to be a (possibly empty) set of f-positions, i.e. a subset of $\{1, \ldots, n\}$.

Given an equation $e : (f\,(c\,y_1 \ldots y_m)\,x_2 \ldots x_n) = r$ in E, and an f-unit u, we define the variable set from e corresponding to u:

$$W^e_u = \{y_j \mid 1 \in u \text{ and } 1 \leq j \leq m\} \cup \{x_i \mid i \in u \text{ and } 2 \leq i \leq n\}$$

E.g. let e_1 and e_2 be the equations for append (in the Introduction), then $W^{e_1}_{\{1\}} = \emptyset, W^{e_2}_{\{1\}} = \{x, y\}$.

In the following, let $\mathcal{F}$ be the family consisting of sets $S^{\mathcal{F}}_f$, where for each function f in the canonical system, $S^{\mathcal{F}}_f$ is a nonempty set of f-units.

For every equation $l = r$ in E: For any particular occurrence of a subterm t of r, define the set $\tau_{\mathcal{F}}(t)$ of *rhs trees for* t by induction on t: $\tau_{\mathcal{F}}(t)$ is the smallest set such that

- If t is a variable then the tree consisting of the single node (labeled) "t" is in $\tau_{\mathcal{F}}(t)$.
- If $t = (c\,t_1 \ldots t_n)$ where c is a constructor, then for any choice of rhs trees $\tau_1, \ldots, \tau_n$ from $\tau_{\mathcal{F}}(t_1), \ldots, \tau_{\mathcal{F}}(t_n)$ respectively, the tree with root c and immediate subtrees $\tau_1, \ldots, \tau_n$ is in $\tau_{\mathcal{F}}(t)$.
- If $t = (g\,t_1 \ldots t_n)$ where g is a function, then for any choice of nonempty g-unit $u = \{i_1, \ldots, i_p\}$ from $S^{\mathcal{F}}_g$ and for any choice of rhs trees $\tau_{i_1}, \ldots, \tau_{i_p}$ from $\tau_{\mathcal{F}}(t_{i_1}), \ldots, \tau_{\mathcal{F}}(t_{i_p})$ respectively, the tree with root g and immediate subtrees $\tau_{i_1}, \ldots, \tau_{i_p}$ is in $\tau_{\mathcal{F}}(t)$.

For a function f, the set of f-units $S^{\mathcal{F}}_f$ is said to be *adequate* if for every equation e for f with rhs r and for every rhs tree $\tau_r \in \tau_{\mathcal{F}}(r)$, when we let V

be the set of variables occurring in τ_r then there is an f-unit $u \in S_f^{\mathcal{F}}$ such that $V \subseteq W_u^e$. We say that u *covers* τ_r. If for every r and τ_r, when $V \neq \emptyset$ there's exactly one f-unit in $S_f^{\mathcal{F}}$ covering τ_r, then $S_f^{\mathcal{F}}$ is said to be *uniquely covering.*

If for every function f, $S_f^{\mathcal{F}}$ is adequate and uniquely covering and contains the position 1, then $\mathcal{F}$ is a *fit family*, $S_f^{\mathcal{F}}$ is a *fit unit set*, the units are *fit units*, and the rhs trees are *fit trees.*

Given an equation $f\,(c\,y_1 \dots y_m)\,x_2 \dots x_n = r$ in E, particular subterms s, t of r such that t is a subterm of s, a rhs tree $\tau_s \in \tau_{\mathcal{F}}(s)$. Define t *is in* τ_s (sometimes written $t \in \tau_s$) if either t and s are the same occurrence of a subterm of r, or if for some constructor or function k, $s = (k\,s_1 \dots s_n)$ and for some immediate subtree τ_{s_i} of τ_s, t is in τ_{s_i}.

Note 2 A unit consisting of all positions is called *a trivial unit* and it is always a fit unit. But often we wish to have smaller fit units. However, for recursive functions finding (small) fit units is a nontrivial task. Future work should be to make "completion algorithms" for handling some cases automatically.

3.1.2 Fit Trees Indicate What Is Needed.

Lemma 1 *Let the following be given: A terminating canonical system, a fit family $\mathcal{F}$, an n-ary function f in the system, ground input $X_1, \dots, X_n$ to f, and let r be the rhs of the equation such that $(f\,\overline{X})$ matches the lhs. Then there's a unique fit tree $\tau_r \in \tau_{\mathcal{F}}(r)$ such that to compute $(f\,\overline{X})$*

- *from the constructors and functions in r we may only need those that occur in τ_r, and*
- *from the input $X_1, \dots, X_n$ we may only need X_i such that $i \in U$, where $U \in S_f^{\mathcal{F}}$ is the fit unit covering τ_r.*

So Lemma 1 states that to compute a function f on input $\overline{X}$ there's a unique fit tree τ_r that contains all that's needed. We will call τ_r *the needed fit tree* and U *the needed fit unit* (w.r.t. f and $\overline{X}$). Note that the lemma only asserts the existence of τ_r, not how to find it, but to us that won't matter.

The fit units can furthermore be used to delimit which recursive calls are needed. We define a tree of recursive calls that only includes those recursive calls that are in needed fit trees:

Definition 1 (fit based tree of recursive calls) Given a terminating canonical system and a fit family $\mathcal{F}$. Let M be an M-set in the system, let $f \in M$

with ground input $X_1, \dots, X_n$. The *fit based tree of recursive calls* $T_{f\overline{X}}$ *for* f *on* $\overline{X}$ is the smallest tree such that

- the root is $(f\, X_1! \dots X_n!)$, and
- for each node $(g\, Y_1 \dots Y_m)$ in $T_{f\overline{X}}$, let r be the rhs of the equation such that $(g\,\overline{Y})$ matches the lhs, let σ be the matching substitution, let the needed fit tree w.r.t. $(g\,\overline{Y})$ be τ_r, then for every subterm $t = (h\, z_1 \dots z_k)$ of r such that t is in τ_r and $h \in M$, $(g\,\overline{Y})$ has a child $(h\,(z_1\sigma)! \dots (z_k\sigma)!)$.

In the quick sort example, we might choose trivial units for all functions except $\{1,2\}, \{1,3\}$ for if-then-else, then in any fit based tree of recursive calls for quicksort every qs-node has at most one child qs. If we didn't delimit by using fit units, the (sub)tree of qs calling itself would have been exponentially large.

3.2 The *poly-basic* System

For a constructor term t, the length of t, written $|t|$, is the number of constructors in t. If the system is terminating then for a ground term t, $|t|$ is the length of t in normal form.

Whenever we say "polynomial", we mean a unary, monotone polynomial function p, i.e. p is a unary function on nonnegative integers of the form $p(x) = a_0 + a_1 x + \cdots + a_k x^k$, where $a_i \geq 0$, $k \geq 0$.

Definition 2 (*poly-basic*) A canonical system with a fit family $\mathcal{F}$ and such that *PC1*[3] and *PC2* hold, is called *poly-basic.*

PC1 For every M-set M there is a polynomial P_M such that for any n-ary function $f \in M$, for any ground terms $X_1, \dots, X_n$, let $U \in S_f^{\mathcal{F}}$ be the needed fit f-unit, let $T_{f\overline{X}}$ be the fit based tree of recursive calls for f on $\overline{X}$: The number of nodes in $T_{f\overline{X}}$ is bounded by $P_M(\sum_{i \in U} |X_i|)$.

PC2 For every M-set M there is a polynomial Q_M and a integer constant C_M such that for any ground term $(f\, X_1 \dots X_n)$ where $f \in M$, let $U \in S_f^{\mathcal{F}}$ be the needed f-unit, let r be the rhs of the equation such that $(f\,\overline{X})$ matches the lhs, let σ be the matching substitution, let τ_r be the needed fit tree, then for any subterm $t = (g\, t_1 \dots t_m)$ of r such that t is in τ_r and the needed g-unit for $t\sigma$ is $V \in S_g^{\mathcal{F}}$:

1. if $g \in M$ then $\sum_{i \in V} |t_i \sigma| \leq \sum_{i \in U} |X_i| + C_M$.

[3]PC means Polynomially Correct

2. if $g \notin M$ then $\sum_{i \in V} |t_i \sigma| \le Q_M(\sum_{i \in U} |X_i|)$.

Note that *PC1* implies termination. quicksort (choosing trivial units except for if-then-else: $\{1,2\},\{1,3\}$) is *poly-basic* - we may e.g. use $P_M(x) = x^2$ for the M-set $M = \{\mathsf{quicksort}, \mathsf{qs}\}$. So by Theorem 1 below, quicksort is poly-time. exp isn't *poly-basic* since in the first equation, *PC2.2* doesn't hold for double's argument.

Theorem 1 (the poly-time theorem) *Let a* poly-basic *system be given. For every M-set M there is a polynomial R_M such that for any n-ary function f, any constructor terms $X_1, \ldots, X_n$, let $U \in S_f^{\mathcal{F}}$ be the needed f-unit for f on $\overline{X}$, then $(f\,\overline{X})$ can be computed in time bounded by $R_M(\sum_{i \in U} |X_i|)$ by using a lazy computation strategy. (And: If fit units are trivial one may use an eager strategy.)*

3.3 Starting to Work on Polynomially Bounded Output Lengths

PC1 and *PC2.1* are trivially satisfied for e.g. function definitions in usual primitive recursive form, i.e. $f\,(c\,\overline{y})\,\overline{x} = h\,\overline{y}\,\overline{x}\,(f\,y_1\,\overline{x}) \ldots (f\,y_m\,\overline{x})$. But *PC2.2* doesn't have such simple subcases. In *poly-basic* we have achieved to express polynomially bounded *computation time* by polynomially bounded *output length* (*PC2.2*). So now we need to study output lengths. Given a terminating system with a fit family, consider *PC3*:

> For every M-set M there is a polynomial Q_M such that for any function $f \in M$, for any ground terms $X_1, \ldots, X_n$, let U be the needed f-unit: $|f\,X_1 \ldots X_n| \le Q_M(\sum_{i \in U} |X_i|)$.

Furthermore consider a function f defined by recursion $\mathsf{f}\,(\mathsf{succ}\,x) = \mathsf{g}\,(\mathsf{f}\,x),\ \mathsf{f}\,0 = 0$, where g is some function satisfying *PC3*. If g satisfies *PC3* with the $Q_g(x) = 2x$ (or greater) but not with any $Q'_g(x) = x + C$ $(C \ge 0)$, then f doesn't satisfy *PC3*. So we learn that recursive calls cannot be "doubled"[4].

To be able to compose functions freely in rhs, the functions should satisfy a stronger *PBO* (Polynomially Bounded Output): The length of the output is bounded

- by a polynomial in the length of the needed input that *doesn't* contain recursive calls, and
- (strictly) linearly (i.e. $p(x) = x + k$, k a constant) by the length of needed input that *does* contain recursive calls.

To formalize *PBO* we will use the concept of critical positions.

[4]It's for a similar reason that we require *PC2.1 inside* recursive calls, otherwise we risk things like $\mathsf{e}\,(\mathsf{succ}\,x)\,y = \mathsf{e}\,x\,(\mathsf{double}\,y),\ \mathsf{e}\,0\,y = y$.

4 Critical Positions

Our idea of defining critical positions for a function derives from works by [1] and [3] to characterize classes of subrecursive functions. There they treat the results of recursive calls (i.e. the recursive call terms) very carefully by distinguishing between different types of arguments to functions: Namely those arguments that may only be used "locally" and bit-wise ([1]'s "safe"); and the rest of the arguments ([1]'s "normal") that may be used "globally" to do recursion on. Any recursive call term used as an argument, is of the first type.

Now our intention is to give a general definition of the critical positions such that they are *exactly* those argument positions that may receive the result of recursive calls, and no others. So the naive definition of a critical position is: Argument position i in function f is critical if in some rhs, f's i'th argument is a recursive call term. E.g. in the definition of quicksort, append's first position is critical since there is a rhs where append's first argument is the recursive call term $(\mathsf{quicksort}\, l)$. But there are two complications about this . The first is that f might have as i'th argument a term t_i which has a *proper subterm* that is a recursive call term (e.g. if append's first argument were $(\mathsf{tail}\,(\mathsf{quicksort}\, l)))$. In this case too, we will define i to be a critical position in f. The second complication has to do with passing arguments from one function to another. We should then "remember" criticality. E.g. redefine exp by

$$\begin{array}{lcllcl}
\mathsf{exp}'\,(\mathsf{succ}\,x) & = & \mathsf{double}'\,(\mathsf{exp}'\,x) & \mathsf{exp}'\,0 & = & \mathsf{succ}\,0 \\
\mathsf{double}'\,(\mathsf{succ}\,x) & = & \mathsf{add}\,(\mathsf{succ}\,x)\,(\mathsf{succ}\,x) & \mathsf{double}'\,0 & = & 0 \\
\mathsf{add}\,(\mathsf{succ}\,x)\,y & = & \mathsf{succ}\,(\mathsf{add}\,x\,y) & \mathsf{add}\,0\,y & = & y
\end{array}$$

The position in double' is critical, so we intend to treat any argument a to double' carefully. double' passes two copies of a over to add, so it's up to add what happens to the a's. We should tell add that both its inputs must be treated carefully, i.e. we let both of add's positions be critical. It's because of this second complication that we will first define critical *variables* and then critical *positions*.

4.1 Definition and Marking Algorithm

Let a canonical system be given. Note that the definition of critical variables and positions is with respect to this system, but to simplify notation, we don't mark this explicitly. Recall the definition of "the variable set from e corresponding to u", W_u^e, from Section 3.1 (one of the first definitions there).

Definition 3 (critical variables in an equation)

- Let $e : lhs = rhs$ be an equation in the given canonical system. If there is a subterm $(f\,t_1 \dots t_m)$ of rhs such that t_i $(1 \leq i \leq m)$ has a subterm which is a critical variable in e or a recursive call term, then this induces that in every equation e' for f: Every $v \in W^{e'}_{\{i\}}$ is a *critical variable in* e'.

- A variable occurring in an equation in the given canonical system, is noncritical if it cannot be demonstrated to be critical.

Consider the definition of $\mathsf{quicksort}$. Since $(\mathsf{quicksort}\,l)$ is a recursive call term so x and y in append are critical. Since $(\mathsf{quicksort}\,r)$ is a recursive call term so z in append is critical. Since $(\mathsf{qs}\,y\,z\,(\mathsf{cons}\,x\,l)\,r))$ and $(\mathsf{qs}\,y\,z\,l\,(\mathsf{cons}\,x\,r))$ are recursive call terms so x and y in $\mathsf{if\text{-}then\text{-}else}$ are critical. Since the variables in $\mathsf{quicksort}$ and qs cannot be demonstrated to be critical, they are noncritical.

Definition 4 (critical positions) For an n-ary function f defined by k equations $e_1, \dots, e_k$: Position i, $1 \leq i \leq n$, is *critical* iff every $v \in (W^{e_1}_{\{i\}} \cup \dots \cup W^{e_k}_{\{i\}})$ is a critical variable.[5]

Note that we haven't said anything about the positions of the constructors.

We will now present a "marking algorithm" for marking all the critical positions in the given canonical system. The algorithm is based on the following observation of Definition 3: Let e, e', f be as in the definition. Say that M is the M-set for which e is an equation, and that N is the M-set for which e' is an equation (so $f \in N$). Then either $N = M$ or $M \rhd N$. In other words, we have observed that the definition only concerns the positions of functions in M and in M-sets below M. For this reason, in the algorithm we can follow the order $\rhd$ downwards, in each M-set M, considering the M-equations and applying the Definition 3 as much as possible.

```
Define the active set $S$ to consist of all the M-sets in the canonical system.
While $S$ isn't empty do
        Let $M$ be a maximal (wrt. $\rhd$) M-set in $S$.
        Repeat
                For every equation $e$ in $M$ do
                    Use definition 3 on $e$ to mark variables as critical.
                od
        until in the last "round", no new critical variable was found in an $M$-equation.
        $M$-variables that aren't critical, are marked as noncritical.
        Remove $M$ from $S$.
od
```

[5]Either all or none of the variables in $W^{e_1}_{\{i\}} \cup \dots \cup W^{e_k}_{\{i\}}$ are critical.

Notation: Given a function f, an f-unit u . We let nu be the subset of u consisting of all the noncritical f-positions and we let cu be the subset of u consisting of all the critical f-positions.

5 Polynomially Bounded Output

5.1 Replacing *PC2.2* by *PBO*

Now we can formalize *PBO* from Section 3.3.

Definition 5 (*PBO* M-set) Let a terminating canonical system with a fit family $\mathcal{F}$ be given. An M-set M in the system satisfies *PBO* if there is a polynomial Q_M such that for any function $f \in M$, for any ground terms $X_1, \ldots, X_n$, when $U = NU \cup CU$ is the needed fit f-unit then

$$|(f\, X_1 \ldots X_n)| \leq Q_M(\sum_{i \in NU} |X_i|) + \sum_{i \in CU} |X_i|$$

Lemma 2 *If in the definition of* poly-basic*, instead of* PC2.2 *we require that* PBO *holds for every M-set, then Theorem 1 still holds.*

The idea of the proof is to separate the construction of polynomials for the "active" noncritical input from the "passive" critical input.

5.2 Motivation for *DDC*

We continue the work on making *PC2.2* more manageable. According to *PBO*, we must investigate how to treat critical arguments linearly.

Say that f on input $X_1, \ldots X_n$ "doubles" its i'th argument if $|f\,\overline{X}| \geq 2|X_i|$. In general there are two ways for a function f to double an argument: Either by doing recursion on the argument, or by writing an equation where the rhs is nonlinear in the variable(s) corresponding to the argument. For instance

$$\begin{array}{lcl} \mathsf{double}_1\,(\mathsf{succ}\,x) & = & \mathsf{succ}\,(\mathsf{succ}\,(\mathsf{double}_1\ x)), \quad \mathsf{double}_1\ 0 \ = \ 0 \\ \mathsf{double}_2\,x & = & \mathsf{cons}\,x\,(\mathsf{cons}\,x\,\mathsf{nil}) \end{array}$$

Consider the first way of doubling arguments. We will forbid all recursion on critical. In *DDC* this will be expressed by *RON* (Recursion On Noncritical), i.e. *PC1* depending only on the noncritical input. So *RON* fits in with the ideas of [1] and [3].

But there's also the second way of doubling, not considered by [1] and [3]. To avoid this kind of doubling, it's not necessary to require that the whole rhs is linear in all critical variables. Rather we should study the use of constructors and make sure that we don't combine several copies of the same critical variables. Formally, we will do this by defining the *output unit*, it's just like the fit unit, but doesn't have to contain position 1. E.g. if-then-else has minimal output units $\{2\}, \{3\}$.

Definition 6 (output family) Let a canonical system be given and a family $\mathcal{F}$ consisting of a nonempty set $S_f^{\mathcal{F}}$ of f-units for each function f. If for every function f, $S_f^{\mathcal{F}}$ is adequate and uniquely covering then $\mathcal{F}$ is an *output family*, $S_f^{\mathcal{F}}$ is an *output unit set*, the units are *output units*, the rhs trees are *output trees*.

We will then require that every output tree is linear in every critical variable. In *DDC* this will be expressed by *LIC* (Linear In Critical).

Note that from now on, for a system we will typically have *two* families of sets of units: A fit family $\mathcal{F}^{\mathcal{F}}$ and an output family $\mathcal{F}^{\mathcal{O}}$.

Intuitively, when recursion is ignored, output units and output trees express what may leave traces in the output; and what may leave traces is decided by the use of constructors and variables. The functions only "conserve" what has been decided.

Consider the important special case of just having nullary and unary constructors (e.g. the *BC* functions). Such constructors cannot combine any inputs. Formally this is expressed by the fact that for n-ary f, we may take the singletons $\{1\}, \ldots, \{n\}$ as output units; then every output tree has only some unary and a nullary branching. The second way of doubling doesn't work, *LIC* is always trivially satisfied.

But what we hope to obtain in this way (in *DDC*) is not exactly *PBO*, but *PBO'* where the length of $(f\, X_1 \ldots X_n)$ depends on the critical *output* positions rather than the critical *fit* positions. To bridge the gap, we will require that output units are chosen to be "unit smaller" than fit units. The technical details of this extra complication about *PBO* and *PBO'* follow in the next subsection.

5.3 Replacing *PBO* by *PBO'*

Analogously to Lemma 1, we have

Lemma 3 (needed when ignoring first arguments) *Let the following be given: A terminating canonical system, an output family $\mathcal{F}$, an n-ary function f in the system, ground input $X_1, \ldots, X_n$ to f, and let r be the rhs of the equation such that $(f\, \overline{X})$ matches the lhs. Assume that the normal form of the first argument of every function called on any input, is* *given*[6]. *Then there's a unique output tree $\tau_r \in \tau_{\mathcal{F}}(r)$ such that to compute $(f\, X_1 \ldots X_n)$*

- *from the constructors and functions in r we may only need those that occur in τ_r, and*
- *from the input $X_1, \ldots, X_n$ we may only need X_i such that $i \in u$, where u is the f-unit covering τ_r.*

We will call τ_r *the needed output tree* and u *the needed output unit* (w.r.t. f, $\overline{X}$).

Definition 7 (*PBO'* M-set) As Definition 5, but add the assumption that there's also an output family $\mathcal{F}^{\mathcal{O}}$ and let the new requirement be that

$$|(f\, X_1 \ldots X_n)| \leq Q_M(\sum_{i \in NU} |X_i|) + \sum_{i \in cu} |X_i|$$

where u is the needed output f-unit for $(f\, \overline{X})$.

Definition 8 ($\mathcal{F}^{\mathcal{O}}$ "unit smaller than" $\mathcal{F}^{\mathcal{F}}$) Let a terminating canonical system be given with a fit family $\mathcal{F}^{\mathcal{F}}$ and an output family $\mathcal{F}^{\mathcal{O}}$. $\mathcal{F}^{\mathcal{O}}$ *is smaller than* $\mathcal{F}^{\mathcal{F}}$ if for every rhs r, every output tree τ_r and every fit tree ρ_r such that τ_r is contained in ρ_r (defined below): If u is the output unit covering τ_r, and U is the fit unit covering ρ_r, then $u \subseteq U$.

Define τ_r is contained in ρ_r by induction on r: Consider a (particular occurrence of a) subterm t of r. Let two rhs trees for t be $\tau_t^O \in \tau_{\mathcal{F}^{\mathcal{O}}}(t)$ and $\tau_t^F \in \tau_{\mathcal{F}^{\mathcal{F}}}(t)$. τ_t^O *is contained in* τ_t^O if the roots in τ_t^O and τ_t^F are the same, and

- if $t = (c\, t_1 \ldots t_n)$ then each $\tau_{t_i}^O$ is contained in $\tau_{t_i}^F$.
- if $t = (g\, t_1 \ldots t_n)$: Let the g-units used be respectively u and U. Then $u \subseteq U$, and for every $i \in u$ we have that $\tau_{t_i}^O$ is contained in $\tau_{t_i}^F$.

Lemma 4 (from *PBO'* to *PBO*) *Let a terminating canonical system with a fit family $\mathcal{F}^{\mathcal{F}}$ and an output family $\mathcal{F}^{\mathcal{O}}$ be given such that $\mathcal{F}^{\mathcal{F}}$ is unit smaller than $\mathcal{F}^{\mathcal{O}}$. Then for every M-set M:*

- *For every $f \in M$, for every input $X_1, \ldots, X_n$ to f: If u is the needed output f-unit and U is the needed fit f-unit, then $u \subseteq U$.*
- *So if M satisfies* PBO' *then M satisfies* PBO *with the same polynomial.*

So if in the definition of poly-basic*, instead of* PC2.2 *we require that* PBO' *holds for every M-set, then Theorem 1 still holds.*

[6]First arguments are not *computed*, not even by an oracle.

6 The "Don't Double Criticals" System (*DDC*)

Let the i'th projection π_i be defined by $\pi_i\,(c\,x_1 \ldots x_m) = x_i$ for every constructor c and $1 \le i \le m$. Let t be a term, let s be a subterm of t. *s is a projection sequence in t* if $s = \pi_{i_1}(\ldots(\pi_{i_n}\,v)\ldots)$ such that $i_n \ge 0$ and v a variable. *s is maximal* if s is t or if the father of π_{i_1} is not a projection.

Definition 9 (*DDC* M-set) Let a canonical system with a fit family $\mathcal{F}^{\mathcal{F}}$ and an output family $\mathcal{F}^{\mathcal{O}}$ be given such that $\mathcal{F}^{\mathcal{F}}$ is unit smaller than $\mathcal{F}^{\mathcal{O}}$. An M-set M is *DDC* if *PC2.1* holds for M and if moreover *RON*, and *LIC* are satisfied:

RON There is a polynomial P_M such that for any n-ary function $f \in M$, for any ground terms $X_1, \ldots, X_n$, let $U = NU \cup CU$ be the needed fit f-unit, let $T_{f\,\overline{X}}$ be the fit based tree of recursive calls for f on $\overline{X}$: The number of nodes in $T_{f\,\overline{X}}$ is bounded by $P_M(\sum_{i \in NU} |X_i|)$.

LIC For every function $f \in M$, for every equation e for f with rhs r, for every output tree τ_r for r: If in τ_r there's a maximal projection sequence $s = \pi_{i_1}(\ldots(\pi_{i_n}\,v)\ldots)$ such that v is a critical variable in e, then there are no other occurrences of the term s in τ_r.

In $\mathsf{quicksort}$, choose trivial fit units and output units except for $\mathsf{if\text{-}then\text{-}else}$: choose $\{1,2\}, \{1,3\}$ and $\{2\}, \{3\}$ respectively. $\mathsf{quicksort}$ is not *DDC* since append recurs on its critical first argument (so *RON* doesn't hold). If we however give append an extra noncritical first argument and simulate the original recursion, this can be fixed (see also $\mathsf{treesort}$ below): Let $\mathsf{C}\,\mathsf{nil}\,a\,b = a, \mathsf{C}\,(\mathsf{cons}\,x\,y)\,a\,b = b$, and let $(\mathsf{length}\,t)$ produce $|t|$ in unary. With the following changes, the system becomes *DDC*.

$$\begin{array}{lcl}
\mathsf{qs}\,\mathsf{nil}\,z\,l\,r & = & \mathsf{append}'\,(\mathsf{length}\,(\mathsf{cons}\,l\,r))\,(\mathsf{quicksort}\,l)\,(\mathsf{cons}\,z\,(\mathsf{quicksort}\,r)) \\
\mathsf{append}'\,(\mathsf{succ}\,n)\,x\,y & = & \mathsf{C}\,x\,y\,(\mathsf{cons}\,(\pi_1\,x)\,(\mathsf{append}'\,n\,(\pi_2\,x)\,y))
\end{array}$$

It's in fact in order to be able to simulate recursion on critical in this way that we bothered to allow projections on critical variables in *LIC*; otherwise *LIC* could simply have required that every critical variable v occurs at most once in τ_r.

Theorem 2 (the output length theorem for *DDC*) *Let a (possibly empty) terminating canonical system with a fit family $\mathcal{F}^{\mathcal{F}}$ and an output family $\mathcal{F}^{\mathcal{O}}$ be given such that $\mathcal{F}^{\mathcal{F}}$ is unit smaller than $\mathcal{F}^{\mathcal{O}}$ and such that* PBO' *holds for every M-set. Define a new* DDC *M-set M using the given system*[7]*. Then* PBO' *is satisfied for M.*

[7]That is, the M-functions may call functions from the given system.

The key ideas in the proof of Lemma 2 are to do a first recursion on the height of the output based tree for $(f\,\overline{X})$, and a second recursion on the structure of the needed rhs. In the second recursion we keep critical input outside the construction of polynomials. The arguments of recursive calls are treated specially with *PC2.1.* By Theorem 2 and (the last two lines in) Lemma 4 we get

Corollary 5 (pure *DDC* is poly-time) *Let a canonical system with a fit family $\mathcal{F}^{\mathcal{F}}$ and an output family $\mathcal{F}^{\mathcal{O}}$ be given such that $\mathcal{F}^{\mathcal{F}}$ is unit smaller than $\mathcal{F}^{\mathcal{O}}$. If every M-set is* DDC *then every function in the system is poly-time.*

So, summing up about the relationships between the different classes: All pure *DDC* functions are *poly-basic* and all *poly-basic* functions are poly-time. In the remaining subsections, we give examples of subsystems of *DDC*. In Appendix 1, we show that all (nonnullary) *BC* functions are pure *DDC*.

6.1 A Very Simple Syntactical Subsystem

As an example of a particularly simple (pure) *DDC* system consider *DDC-simple1*: A canonical system (with trivial output units and fit units) satisfying

1. The first position of every recursive function is noncritical.
2. Every recursive call term has the form $(g\,v_1 \dots v_n)$ where the v_i's are different variables.
3. For every M-set M, for every equation of the form

   $$f\,(c\,y_1 \dots y_m)\,x_2 \dots x_n = \cdots (g_1\,v_{1,1} \dots v_{1,r_1}) \cdots (g_k\,v_{k,1} \dots v_{k,r_k})$$

   where $f, g_1, \dots, g_k$ all are in M and $k \geq 2$, either

   (a) Any $(g_i\,v_{i,1} \dots v_{i,r_i})$ and $(g_j\,v_{j,1} \dots v_{j,r_j})$, $i \neq j$, have no variables in common, or

   (b) Each $v_{j,1}$ is some y_p, and all the $v_{1,1}, \dots, v_{k,1}$ are different.
4. Every rhs is linear in every critical variable.

Note that Condition 4 is very restrictive. However, unary addition and multiplication are *DDCsimple1*: $\mathsf{mul}\,0\,y = 0$, $\mathsf{mul}\,(\mathsf{succ}\,x)\,y = \mathsf{add}\,y\,(\mathsf{mul}\,x\,y)$, $\mathsf{add}\,0\,y = y$, $\mathsf{add}\,(\mathsf{succ}\,x)\,y = \mathsf{succ}\,(\mathsf{add}\,x\,y)$.

6.2 A Little More Complicated Subsystem

As a second example of a subsystem of (pure) *DDC*, consider *DDCsimple2*: Let a canonical system with a fit family $\mathcal{F}^{\mathcal{F}}$ and an output family $\mathcal{F}^{\mathcal{O}}$ be given such that $\mathcal{F}^{\mathcal{F}}$ is unit smaller than $\mathcal{F}^{\mathcal{O}}$. The system is *DDCsimple2* if *LIC* and the following conditions are satisfied.

1. The first position of every recursive function is noncritical.
2. Every recursive call term has the form $(g\, t_1 \ldots t_m)$ where each t_i is a maximal projection sequence; and for every fit tree $\tau_{g\,\bar{t}}$ for $(g\,\bar{t})$: If there's a maximal projection sequence s in $\tau_{g\,\bar{t}}$, then there are no other occurrences of the term s in $\tau_{g\,\bar{t}}$.
3. For every equation $f\,(c\, y_1 \ldots y_m)\, x_2 \ldots x_n = r$ we have that: For every recursive call term $(g\, t_1 \ldots t_k)$ in r we have that t_1 is a y_i; and for any fit tree τ_r for r, for any two recursive call terms $(g\, y_i t_2 \ldots t_m)$ and $(h\, y_j s_2 \ldots s_n)$ in τ_r, we have that $i \neq j$.

Example 1 quicksort isn't *DDCsimple2* for the same reason as it wasn't *DDC* and because arguments in recursive call terms contain constructors, and because in qs's first equation, quicksort is called (twice) with illegal first argument.

Example 2 In the definition of tree sort below, we use intermediate binary trees constructed from ternary bic (value, left subtree, right subtree) and nullary emp. The critical positions are the first position in insert, the second and third positions in if-then-else, both positions in append. Choose trivial fit units and output units except for if-then-else ($\{1,2\}, \{1,3\}$ and $\{2\}, \{3\}$, as usually).

$$
\begin{array}{lcl}
\mathsf{treesort}\, l & = & \mathsf{flatten}\,(\mathsf{maketree}\, l\,) \\
\mathsf{maketree}\,\mathsf{nil} & = & \mathsf{emp} \\
\mathsf{maketree}\,(\mathsf{cons}\, x\, y) & = & \mathsf{insert}\,(\mathsf{maketree}\, y)\, x \\
\mathsf{insert}\,\mathsf{emp}\, x & = & \mathsf{bic}\, x\, \mathsf{emp}\, \mathsf{emp} \\
\mathsf{insert}\,(\mathsf{bic}\, v\, l\, r)\, x & = & \mathsf{if\text{-}then\text{-}else}\,(\mathsf{lesseq}\, x\, v)\,(\mathsf{bic}\, v\,(\mathsf{insert}\, l\, x)\, r) \\
 & & (\mathsf{bic}\, v\, l\,(\mathsf{insert}\, r\, x)) \\
\mathsf{flatten}\,\mathsf{emp} & = & \mathsf{nil} \\
\mathsf{flatten}\,(\mathsf{bic}\, v\, l\, r) & = & \mathsf{append}\,(\mathsf{flatten}\, l)\,(\mathsf{cons}\, v\,(\mathsf{flatten}\, r))
\end{array}
$$

treesort isn't *DDCsimple2* since append and insert have critical first arguments. As for quicksort, this can be fixed: Let $\mathsf{C}\,\mathsf{emp}\, a\, b = a$, $\mathsf{C}\,(\mathsf{bic}\, x\, y\, z)\, a\, b = b$, and let $(\mathsf{triplelength}\, t)$ produce $3|t|$ in unary. With the following changes, the system becomes *DDCsimple2* (append' and length are as when we "fixed" quicksort).

$$\begin{array}{lcl}
\mathsf{maketree}\,(\mathsf{cons}\,x\,y) & = & \mathsf{insert}'(\mathsf{triplelength}\,y)\;(\mathsf{maketree}\,y)\;x \\
\mathsf{insert}'\,(\mathsf{succ}\,n)\,t\,x & = & \mathsf{C}\,t\,(\mathsf{bic}\,x\,\mathsf{emp}\,\mathsf{emp})\;\;(\mathsf{if\text{-}then\text{-}else}\;(\mathsf{lesseq}\,x\,(\pi_1\,t)) \\
 & & (\mathsf{bic}\,(\pi_1\,t)\;(\mathsf{insert}'\,n\,(\pi_2\,t)\,x)\;(\pi_3\,t))) \\
 & & (\mathsf{bic}\,(\pi_1\,t)\;(\pi_2\,t)\;(\mathsf{insert}'\,n\,(\pi_3\,t)\,x)) \\
\mathsf{flatten}\,(\mathsf{bic}\,b\,l\,r) & = & \mathsf{append}'\,(\mathsf{length}\,(\mathsf{bic}\,b\,l\,r))\;(\mathsf{flatten}\,l)\;(\mathsf{cons}\,b\,(\mathsf{flatten}\,r))
\end{array}$$

Example 3 f receives a list of binary strings and outputs the list where those elements that didn't contain a string with an "s_1" in it, have been removed. Choose the fit g-units as $\{1,2\},\{1,3\}$, choose output g-units as $\{2\},\{3\}$; choose trivial output and fit units for f. Then f and g are *DDC-simple2*.

$$\begin{array}{lcl}
\mathsf{f}\,\mathsf{nil} & = & \mathsf{nil},\;\; \mathsf{f}\,(\mathsf{cons}\,x\,l) \;=\; \mathsf{g}\,x\,(\mathsf{cons}\,x\,(\mathsf{f}\,l))\,(\mathsf{f}\,l) \\
\mathsf{g}\,\epsilon\,a\,b & = & b,\;\; \mathsf{g}\,(\mathsf{s}_0\,x)\,a\,b = \mathsf{g}\,x\,a\,b,\;\; \mathsf{g}\,(\mathsf{s}_1\,x)\,a\,b = a
\end{array}$$

7 Conclusion

We have studied equations defining functions on constructor data structures and tried to single out some characteristics of equations defining poly-time functions. To this purpose we analyzed *where* the result of a recursive call might be available, and called these argument positions critical. Our criticality analysis is done *after* equations have been written down, whereas in [1] and [3], writing equations and analyzing equations are mixed up

We showed that the concept of criticality is a useful syntactical tool in identifying computationally hard points in function definitions: The point is that a critical argument should not be doubled (cf. exp in the Introduction). In the proposed poly-time system *DDC*, the two ideas how to avoid doubling, were *RON* - recur on noncritical; and *LIC* - use every critical variable only once in the output. The idea of *RON* is already present in [1] and [3], but *LIC* is entirely ours. In [1], *LIC* wasn't required since constructors have arity less than two. In [3], there's nothing like *LIC* and in fact, poly-time is generally lost.

In principle, any poly-time function can be defined in pure *DDC* since the *BC* functions are *DDC* (see proof of this in Appendix 1). But a direct characterization of the poly-time functions on any data structure would be more satisfactory; we are working on the problem. In future we also wish to abandon *PC2.1* and study tail recursion.

References

[1] S. Bellantoni, S. Cook: *A new recursion-theoretic characterization of the polytime functions*, 24th Annual ACM STOC, 1992, 283-293

[2] V.-H. Caseiro: *Some general criteria on equations to guarantee poly-time functions*, Research report no 224, ISBN 82-7368-138-6, ISSN 0806-3036, University of Oslo, Department of Informatics.

[3] D. Leivant: *Stratified functional programs and computational complexity*, 20th ACM Symposium on Principles of Programming Languages, 1993

Now two appendices containing proofs follow. The first shows that the Bellantoni-Cook functions are definable in pure *DDC*, and the second contains the proofs of all the results in this report.

8 Appendix 1: The Bellantoni-Cook Functions Are *DDC*

In [1], Bellantoni and Cook (*BC*) give an equational system in which every function on binary strings[8] can be defined. For every function f, *BC* distinguish between two kinds of argument positions (or input), safe and normal, and they write $(f\,\overline{x};\overline{a})$ to separate normal (on the left) from safe.

Definition 10 (the *BC* functions, from [1]) There are the following constructors: Nullary empty string ϵ and the unary successors s_0 and s_1. In the rhs, s_0 and s_1 may only be applied to safe arguments. The following functions are *BC* functions:

$$\begin{array}{llll} \mathsf{p}\,;\epsilon=\epsilon, & \mathsf{p}\,;(\mathsf{s}_0\,a)=a, & \mathsf{p}\,;(\mathsf{s}_1\,a)=a & \text{predecessor} \\ \mathsf{C}\,;\epsilon\,b\,c\,d=b, & \mathsf{C}\,;(\mathsf{s}_0\,a)\,b\,c\,d=c, & \mathsf{C}\,;(\mathsf{s}_1\,a)\,b\,c\,d=d & \text{conditional} \\ \multicolumn{3}{l}{\mathsf{p}_i^{m,n}\,x_1\ldots x_m;x_{m+1}\ldots x_{m+n}=x_i} & \text{projections} \end{array}$$

There's the following recursion scheme: Given *BC* functions or constructors g, h_{s_i}, define a *BC* function f ($\overline{x}$ and $\overline{a}$ are sequences of variables, i is 0 or 1)

$$\begin{array}{lcl} f\,\epsilon\,\overline{x};\overline{a} & = & g\,\overline{x};\overline{a} \\ f\,(\mathsf{s}_i\,y)\,\overline{x};\overline{a} & = & h_{s_i}\,y\,\overline{x};\overline{a}(f\,y\,\overline{x};\overline{a}) \end{array}$$

There's the following composition scheme: Given *BC* functions or constructors $h, \overline{r}, \overline{t}$, define a *BC* function f ($\overline{x}$ and $\overline{a}$ are sequences of variables)

$$f\,\overline{x};\overline{a}=h\,(\overline{r}\,\overline{x};);(\overline{t}\,\overline{x};\overline{a})$$

[8] Actually they deal with binary numbers and they don't use the word of "constructor".

Lemma 6 *Any set of nonnullary* BC *functions is a* DDCsimple2 *system.*

Proof of Lemma 6 Observe that we may regard any set of *BC* equations as a canonical system by removing ";" and by remembering that to us, composition is shorthand notation. The original *nullary BC* functions are lost in the canonical system.

Let a set of *BC* functions be given (considered as a canonical system). For every n-ary function f, choose all singleton units $\{1\}, \ldots, \{n\}$ as output units and choose the trivial unit fit unit (as we may generally do when all constructors have arity less than two, see Section 5.2). Note that then every output tree has only some unary and a nullary branching, so *LIC* is obviously satisfied. As for the three other requirements to *DDCsimple2*:

1. The first position of any recursive function is noncritical by Lemma 7.
2. The arguments in every recursive call term $(f\, y\, \overline{x}; \overline{a})$ are different variables.
3. The first argument in every recursive call term $(f\, y\, \overline{x}; \overline{a})$ is y, ok. There is maximally one recursive call term in any rhs.

○

Lemma 7 *Let a set of* BC *functions be given. For every function f, every position i in f: If i is normal then i is noncritical.*

Remark 1 The lemma implies that "If a position is critical then it is safe". The opposite direction is not true for the initial functions and not in general because one can *choose* to let positions be safe.

Proof of Proposition 7 Let a set of *BC* functions be given. Observe that in a *BC* system, assignment of critical/noncritical to a function f is completely decided by the functions different from f that call f. We proceed by downward induction on the M-sets.

For a function that isn't called by any other function, except possibly itself, all positions are noncritical.

We wish to show the lemma for a function q such that there's a nonempty set E of equations where q occurs in the rhs, but not in the lhs. Let i be a normal position in q. Then by inspection and by induction hypothesis, we have that in all the equations in E the i'th argument of q neither contains a recursive call term nor a critical variable. So i is noncritical. ○

9 Appendix 2: Proofs of the Results in This Report

Definition 11 (output based tree of recursive calls) As Definition 1, but with an output family $\mathcal{F}$ instead.

9.1 Proofs of Lemmas from Section 5

Various definitions: Let a canonical system be given. A variable v is *loose* in a rhs r in the M-set M if when we think of r as a tree, v has an occurrence in r without any function $f \in M$ above it. $LCV(\tau)$ is the set of loose, critical variables in a rhs tree τ. For a variable v, s, $\#_o(v, \tau)$ is the number of occurrences of v in τ.

Definition 12 (polynomials $z(t, l)$) Let a canonical system with a fit family $\mathcal{F}$ be given. Choose an M-set M in the system, and assume that the system is such that for every M-set below M, *PBO* is satisfied. Define z : (subterm of M-rhs) $\times$ (natural number variable l) $\to$ (polynomial in l) by induction on its first argument:

$$\begin{array}{lcl} z(v, l) & = & l, \text{ for } v \text{ a noncritical variable} \\ z(v, l) & = & 0, \text{ for } v \text{ a critical variable} \\ z(c\, t_1 \ldots t_m, l) & = & 1 + z(t_1, l) + \cdots + z(t_m, l), \text{ for } c \text{ a constructor} \\ z(g\, t_1 \ldots t_m, l) & = & 0, \text{ for } g \in M \\ z(g\, t_1 \ldots t_m, l) & = & Q_{M_g}(\sum_{i\ noncrit} z(t_i, l)) + \sum_{i\ crit} z(t_i, l), \text{ for } g \notin M \end{array}$$

where Q_{M_g} is the polynomial for M_g given by *PBO*.

Proof of Lemma 2 It's enough to prove Lemma 8 below. ○

Lemma 8 (*PBO* gives polynomially bounded subterms) *Let a terminating canonical system with a fit family $\mathcal{F}$ be given such that for every M-set,* PC2.1 *and* PBO *hold. Then for every M-set M there is a polynomial D_M such that for any ground term $(f\, X_1 \ldots X_n)$ where $f \in M$, let $U \in S_f^{\mathcal{F}}$ be the needed fit f-unit, let r be the rhs of the equation such that $(f\, \overline{X})$ matches the lhs, let σ be the matching substitution, let τ_r be the needed fit tree, then for any subterm t of r such that t is in τ_r:*

$$|t\, \sigma| \leq D_M(\sum_{i \in U} |X_i|)$$

Proof of Lemma 8 By upward induction on M-sets. Now we are in an M-set M. We wish to construct a polynomial D_M.

Lemma 9 says that for any $f \in M$, any input input $X_1, \dots, X_n$, when we let τ be the needed fit f-tree, we let $q_1, \dots, q_k$ be the "outermost" recursive call terms in τ_t, we let $U = NU \cup CU$ the needed fit f-unit, we let $L = \sum_{i \in U} |X_i|$, $L' = \sum_{i \in NU} |X_i|$; then for any t in τ, except a proper subterm of $q_1, \dots, q_k$, we have

$$|t\sigma| \leq z(t, L') + \sum_{v \in LCV(\tau_t)} \#_o(v, \tau_t)|v\sigma| + |q_1\sigma| + \cdots + |q_k\sigma|$$

We have that

- $z(t, L') \leq z(t, L)$ since z is monotone and $L' \leq L$.
- $\sum_{v \in LCV(\tau_t)} \#_o(v, \tau_t)|v\sigma| \leq kL$ for some constant k since U covers τ.
- By *PBO* and by *PC2.1* and property of polynomials, each $|q_j\sigma|$ is bounded by a polynomial in L.

So for every subterm $t \in \tau$ (by *PC2.1* and coverings, also for t argument to recursive call), there is some polynomial pol_t such that $|t\sigma| \leq pol_t(L)$.

Define the polynomial $D_M(l)$ to be the sum of $pol_t(l)$ for every subterm t of all fit trees for rhs's in M. Then obviously D_M is as wished. $\bigcirc$

The following lemma is used in the proof of Lemma 8.

Lemma 9 (length of term in fit tree) *Let a terminating canonical system with a fit family $\mathcal{F}$ be given such that for every M-set* PBO *holds. For any M-set M in the system, for any n-ary $f \in M$ with input $X_1, \dots, X_n$: When we let σ be the substitution such that $(f\, X_1 \dots X_n)$ matches the lhs of the equation with rhs r, we let τ_r be the needed fit f-tree, we let U be the needed fit f-unit, we let $L = \sum_{i \in \mathrm{NU}} |X_i|$, we let $q_1, \dots, q_k$ be the recursive call terms in τ_t such that in r there's no $h \in M$ above (outside) them, then for any subterm t in τ_r, except an argument to a $q_1, \dots, q_k$, we have*

$$|t\sigma| \leq z(t, L) + \sum_{v \in \mathrm{LCV}(\tau_t)} \#_o(v, \tau_t)|v\sigma| + |q_1\sigma| + \cdots + |q_k\sigma|$$

Proof of Lemma 9

We prove Lemma 9 by induction on the structure of t. If t is

- A loose variable or a recursive call term without any $h \in M$ above it, it's ok.
- $(c\, t_1 \dots t_m)$: By induction hypothesis for $t_j, 1 \leq j \leq m$

$$\begin{array}{rcl} |t\sigma| & \leq & 1 + |t_1\sigma| + \cdots + |t_m\sigma| \\ & = & z(c\, t_1 \dots t_m, L) + \sum_{v \in LCV(\tau_t)} \#_o(v, \tau_t)|v\sigma| + |q_1\sigma| + \cdots + |q_k\sigma| \end{array}$$

- $(g\,t_1 \ldots t_m), g \notin M$: By *PBO* for M_g, there's a polynomial Q_{M_g} for M_g such that for the needed fit g-unit V, where $V = NV \cup CV$.

  $$|(g\,t_1 \ldots t_m)\sigma| \le Q_{M_g}(\sum_{j \in NV} |t_j\sigma|) + \sum_{j \in CV} |t_j\sigma|$$

  (Note: Here's important that Q_{M_g} only gets noncritical arguments.) We have

  - for $j \in NV$: By induction hypothesis, $|t_j\sigma| \le z(t_j, L)$. (Notice: There aren't any recursive calls, nor critical variables in these t_j's.) So
    $$\sum_{j \in NV} |t_j\sigma| \le \sum_{j\ noncrit} z(t_j, L)$$
  - for $j \in CV$ by induction hypothesis for each t_j, we get

    $\sum_{j \in CV} |t_j\sigma|$
    $\le \sum_{i\ crit} z(t_i, L) + \sum_{v \in LCV(\tau_t)} \#_o(v, \tau_t)|v\sigma| + |q_1\sigma| + \cdots + |q_k\sigma|$

  Altogether

  $$\begin{array}{rcl} |t\sigma| & \le & Q_{M_g}(\sum_{j\ noncrit} z(t_j, L)) + \sum_{i\ crit} z(t_i, L) + \\ & & \sum_{v \in LCV(\tau_t)} \#_o(v, \tau_t)|v\sigma| + |q_1\sigma| + \cdots + |q_k\sigma| \\ & = & z(g\,t_1 \ldots t_m, L) + \sum_{v \in LCV(\tau_t)} \#_o(v, \tau_t)|v\sigma| + |q_1\sigma| + \cdots + |q_k\sigma| \end{array}$$

○

Proof of Lemma 3 Let $f \in M$ with input $X_1, \ldots, X_n$ be given with appropriate substitution σ and with appropriate equation e with rhs r. We proceed by induction on the longest call sequence of functions starting with f on $X_1, \ldots, X_n$. Basis and step are proved in the same way.

We will show that: For every subterm t of r, there's a unique output tree $\tau_t \in \tau_{\mathcal{F}}(t)$ such that if the first argument of every function called in the computation of $t\sigma$ is given, to compute $t\sigma$, from the constructors and functions in t we only need those that occur in τ_t, and from $X_1, \ldots, X_n$ we only need X_i such that there is a variable $v \in W^e_{\{i\}}$ such that $v \in \tau_t$.

We proceed by induction on t.

- If t is a variable, let τ_t consist of the single node labeled t.
- If t is $(c\,t_1 \ldots t_m)$ where c is a constructor, then by induction hypothesis there are $\tau_{t_1}, \ldots, \tau_{t_m}$ as wished, so let τ_t be the tree with root c and immediate subtrees $\tau_{t_1}, \ldots, \tau_{t_m}$.

- If t is $(g\, t_1 \dots t_m)$ where g is a function, then to compute $t\sigma$ when first arguments are given, by induction hypothesis (since g's call sequence is shorter) the lemma holds so there's a unique g-unit u_g such that from the input $t_1\sigma, \dots, t_m\sigma$ we may only need $t_j\sigma$ such that $j \in u_g$. For each $t_j, j \in U_g$, by induction hypothesis there is a τ_{t_j}, so that to compute $t_j\sigma$, when first arguments are given, from the constructors and functions in t_j we only need those that occur in τ_{t_j}, and from $X_1, \dots, X_n$ we only need X_i such that there is a variable $v \in W^e_{\{i\}}$ such that $v \in \tau_{t_j}$. So let τ_t be the tree with root g and as immediate subtrees the τ_{t_j}'s. Then τ_t is a rhs tree as wished.

Then to compute $(f\, X_1 \dots X_n)$ we need X_1 and the variables $X_i, i \geq 2$ such that some corresponding variable occurs in the uniquely chosen output tree τ_r. We have assumed that X_1 is given and by definition of output unit sets, we know that there's a unique output f-unit covering τ_r. ◯

Proof of Lemma 4 We show the first part: For every $f \in M$, for every input $X_1, \dots, X_n$ to f: If u is the needed output f-unit and U is the needed fit f-unit, then $u \subseteq U$. By upward induction on M-sets, basis and step the same way. Consider an M-set M, $f \in M$, input $X_1, \dots, X_n$. By induction on the height of the output based tree T of recursive calls on $(f\, X_1 \dots X_n)$ we show the more general result that: For every node $(h\, Z_1 \dots Z_m)$ in T: If u_h and U_h are the needed output and fit units respectively, then $u_h \subseteq U_h$. Basis and step are in the same way. Fix $(h\, Z_1 \dots Z_m)$, u_h, U_h, appropriate rhs r_h, appropriate substitution σ_h.

We show that: For every subterm t of r_h: If t is in both τ_{r_h} and ρ_{r_h} then for the subtrees τ_t and ρ_t we have that τ_t is contained in ρ_t. Induction on t.

- If t is a variable: Then τ_t and ρ_t are identical, ok.
- If $t = (c\, t_1 \dots t_n)$: Then it's ok by the induction hypothesis for the t_i's.
- If $t = (g\, t_1 \dots t_n)$, $g \notin M$: Then the first part of Lemma 4 holds for $t\sigma_h$, so for the needed units u_g, U_g we have $u_g \subseteq U_g$, and by induction hypothesis for $i \in u_g$, τ_{t_i} is contained in ρ_{t_i}, so ok.
- If $t = (g\, t_1 \dots t_n)$, $g \in M$: By induction hypothesis we have $u_g \subseteq U_g$ so ok again.

So since r_h is in τ_{r_h} and in ρ_{r_h} we have that τ_{r_h} is contained in ρ_{r_h}. By unit smallness $u_h \subseteq U_h$. ◯

9.2 Proof of Theorem 2

Let a canonical system be given. Whenever we talk about a subterm $(\overline{\pi}\, v) = \pi_{i_1}(\dots(\pi_{i_n}\, v)\dots)$ of some rhs r in some M-set M in the system, we mean that $(\overline{\pi}\, v)$ is a maximal projection sequence and that v is critical. Define $(\overline{\pi}\, v)$ in r to be *loose* if (when we think of r as a tree) $(\overline{\pi}\, v)$ doesn't have any M-function above it in r.

Definition 13 (polynomials $w(t, l)$) Let a canonical system with a fit family $\mathcal{F}^{\mathcal{F}}$ and an output family $\mathcal{F}^{\mathcal{O}}$ be given. Let M be a *DDC* M-set in the system such that for every M-set below M, *PBO'* is satisfied. Define w : (subterm of M-rhs) $\times$ (natural number variable l) $\rightarrow$ (polynomial in l) by induction on its first argument:

$$\begin{array}{lcl}
w(v, l) & = & l, \text{ for } v \text{ a noncritical variable} \\
w(v, l) & = & 0, \text{ for } v \text{ a critical variable} \\
w(c\, t_1 \dots t_m, l) & = & l + w(t_1, l) + \dots + w(t_m, l), \text{ for } c \text{ a constructor} \\
w(g\, t_1 \dots t_m, l) & = & lP_M(l + A_M)(\sum_{r \text{ } rhs \text{ } in \text{ } M} w(r, a) + \kappa a) + \sum_{i \text{ } crit} w(t_i, l + A_M), \text{ for } g \in M \\
w(g\, t_1 \dots t_m, l) & = & lQ_{M_g}(\sum_{i \text{ } noncrit} w(t_i, l)) + \sum_{i \text{ } crit} w(t_i, l), \text{ for } g \notin M
\end{array}$$

where $a = l + A_M P_M(l + A_M) + A_M$, κ is the maximal number of recursive calls in any rhs in M, P_M is from *RON*, A_M is the sum of C_M (from *PC2.1*) and the maximal number of critical arguments to any M-function, Q_{M_g} is the polynomial for M_g given by *PBO'*.

Observe that for every t, $w(t, l)$ is a (monotone) polynomial with constant part 0 (i.e. $w(t, l) = a_0 + a_1 l + \dots + a_k l^k$ for $a_0 = 0$ and some $a_i \geq 0$, $k \geq 0$.

It's enough to prove Lemma 10 since then we may define $Q_M(x) = \sum_{r \text{ } rhs \text{ } in \text{ } M} w(r, x)$ and by Part 1 we get (for the appropriate rhs r and needed output tree τ_r)

$$\begin{array}{rcl}
|(f\, X_1 \dots X_n)| & \leq & w(r, \sum_{i \in NU} |X_i|) + \sum_{(\overline{\pi}\, x_i) \in \tau_r} |\overline{\pi}\, X_i| \\
 & \leq & Q_M(\sum_{i \in NU} |X_i|) + \sum_{i \in cu} |X_i|
\end{array}$$

Remark 2 Originally, we tried only to prove Part 2 of Lemma 10 (which gives Theorem 2). But then we weren't able to show Equation 1 below (in two and a half pages). So we added Part 1. And originally the definition of w was just like the definition of z (Definition 12) except in the case that $g \in M$, but then (also because of Equation 1) we found it convenient to have a w with constant 0. As it is now, the proof is perhaps unnecessary long, e.g. we would like to merge Part1 and the auxiliary Lemma 11.

Lemma 10 *Let a (possibly empty) terminating canonical system with a fit family $\mathcal{F}^{\mathcal{F}}$ and an output family $\mathcal{F}^{\mathcal{O}}$ be given such that $\mathcal{F}^{\mathcal{F}}$ is unit smaller than $\mathcal{F}^{\mathcal{O}}$ and such that* PBO' *holds for every M-set. Define a new* DDC *M-set M using the given system.*

Fix T to be an output based tree of recursive calls for some M-function on some input. Let $(f\, X_1 \dots X_m)$ be a node in T deciding a subtree f-tree, *with needed fit unit U_f and needed output unit u_f, let the needed output f-tree* *be τ_r. Let $(f\,(c\,y_1 \dots y_m)\,x_2 \dots x_n) = r$ be the appropriate equation, σ the appropriate substitution. Then*

Part 1 *For every t in τ_r except a nonmaximal projection sequence:*

$$|t\,\sigma| \le w(t, \mathrm{NL}_f) + \sum_{(\overline{\pi}\,x_i) \in \tau_t} |\overline{\pi}\,X_i|$$

Part 2

$$|f\,X_1 \dots X_m| \le |f\text{-tree}|\Big(\sum_{r \text{ rhs in } M} w(r,b) + \kappa b\Big) + \sum_{(\overline{\pi}\,x_i) \in \tau_r} |\overline{\pi}\,X_i|$$

where $NL_f = \sum_{i \in \mathrm{NU}_f} |X_i|$, $b = NL_f + A_M |f\text{-tree}|$.

Proof of Lemma 10 By induction on the height of *f-tree*. The structure is: First we prove Part 1, using induction hypothesis for Part 2. Then we prove the second part, using Part 1 and also the induction hypothesis for Part 2.
Part 1: Basis and step in the same way. By induction on t. If t is

- A noncritical variable: Ok.
- A maximal projection sequence: Ok.
- $(c\,t_1 \dots t_m)$: By induction hypothesis for $t_j, 1 \le j \le m$ and *LIC*

$$\begin{array}{rcl} |t\sigma| & \le & 1 + |t_1\sigma| + \dots + |t_m\sigma| \\ & \le & 1 + w(t_1, L) + \dots + w(t_m, L) + \sum_{i=1}^{m} \sum_{\overline{\pi}\,x_j \in \tau_{t_i}} |\overline{\pi}\,X_j| \\ & = & w(t, NL_f) + \sum_{\overline{\pi}\,x_j \in \tau_t} |\overline{\pi}\,X_j| \end{array}$$

- $(g\,t_1 \dots t_m)$, where $g \notin M$:
  PBO' holds for M_g, so there's a polynomial Q_{M_g} for M_g such that for the needed output g-unit v and for the needed fit g-unit V

$$|(g\,t_1 \dots t_m)\sigma| \le Q_{M_g}\Big(\sum_{j \in NV} |t_j\sigma|\Big) + \sum_{j \in cv} |t_j\sigma|$$

  (Note: Here's important that Q_{M_g} only gets noncritical arguments.) We have

  - for $j \in NV$: By Lemma 12 (not by induction hypothesis), $|t_j\sigma| \le w(t_j, NL_f)$. So

$$\sum_{j \in NV} |t_j\sigma| \le \sum_{j \in NV} w(t_j, NL_f) \le \sum_{j \text{ noncrit}} w(t_j, NL_f)$$

– for $j \in cv$ we have by induction hypothesis for t_j and by *LIC*

$$\sum_{j \in cv} |t_j \sigma| \leq \sum_{i\ crit} w(t_i, NL_f) + \sum_{\overline{\pi}\, x_i \in \tau_t} |\overline{\pi}\, X_i|$$

Altogether

$$\begin{array}{rcl} |t\sigma| & \leq & Q_{M_f}(\sum_{j \in\ noncrit} w(t_j, NL_f)) + \sum_{i\ crit} w(t_i, NL_f) + \sum_{\overline{\pi}\, x_i \in \tau_t} |\overline{\pi}\, X_i| \\ & \leq & w(t, NL_f) + \sum_{\overline{\pi}\, x_i \in \tau_t} |\overline{\pi}\, X_i| \end{array}$$

- $(g\, t_1 \ldots t_m)$, where $g \in M$: Let v be the needed output g-unit, V the needed fit g-unit, τ_{r_g} the needed output tree for g's appropriate rhs r_g. and let $NL_g = \sum_{i \in NV} |t_i \sigma|$. By *RON* and unit smallness we have $|g\text{-}tree| \leq P_M(NL_g)$. By *PC2.1* and since NL_g doesn't depend on critical arguments to f, we have $NL_g \leq NL_f + A_M$. By induction hypothesis part two for g-*tre*, *LIC* for τ_g, induction hypothesis for t_i's in τ_t, *LIC* for τ_t we get

$$\begin{array}{rcl} |t\sigma| & \leq & |g\text{-}tree|(\sum w(r, b_g) + \kappa b_g) + \sum_{\overline{\pi}\, z_i \in \tau_{r_g}} |\overline{\pi}\, t_i \sigma| \\ & \leq & P_M(NL_g)(\sum w(r, B_g) + \kappa B_g) + \sum_{i \in cv} |t_i \sigma| \\ & \leq & P_M(NL_g)(\sum w(r, B_g) + \kappa B_g) + \sum_{i \in cv}(w(t_i, NL_g) + \sum_{\overline{\pi}\, x_j \in \tau_{t_i}} |\overline{\pi}\, X_j|) \\ & \leq & P_M(NL_f + A_M)(\sum w(r, B'_g) + \kappa B'_g) + \\ & & \sum_{i\ crit} w(t_i, NL_f + A_M) + \sum_{\overline{\pi}\, x_j \in \tau_t} |\overline{\pi}\, X_j| \\ & \leq & w(t, NL_f) + \sum_{\overline{\pi}\, x_j \in \tau_t} |\overline{\pi}\, X_j| \end{array}$$

  where $b_g = NL_g + A_M |g\text{-}tree|$, $B_g = NL_g + A_M P_M(NL_g)$, $B'_g = NL_f + A_M P_M(NL_f + A_M) + A_M$.

Part 2: In the induction basis we have that in r there aren't any recursive call terms. By Part 1 of this lemma:

$$\begin{array}{rcl} |f\, X_1 \ldots X_m| & = & |r\sigma| \\ & \leq & w(r, NL_f) + \sum_{\overline{\pi}\, x_i\ in\ \tau_r} |\overline{\pi}\, X_i| \\ & \leq & w(r, b) + \sum_{\overline{\pi}\, x_i\ in\ \tau_r} |\overline{\pi}\, X_i| \end{array}$$

We move on to the inductive step. Consider those recursive call terms in r that don't have M-functions above (outside) themselves. Among these, let $(g_1\, Z_{1,1} \ldots Z_{1,a_1})$, ..., $(g_k\, Z_{k,1} \ldots Z_{k,a_k})$ be those that are children of $(f\, X_1 \ldots X_m)$ in T ($k \geq 1$). So the shape of the equation for f is $f \cdots = \cdots (g_1\, z_{1,1} \ldots z_{1,a_1}) \cdots (g_k\, z_{k,1} \ldots z_{k,a_k} \cdots)$. Assume that the j'th child needs fit unit U_j, output units u_j, equation $l_j = r_j$, output tree τ_{r_j}. Let $NL_j = \sum_{i \in NU_j} |Z_{j,i}|$. By induction hypothesis for g_j:

$$|g_j\, Z_{j,1} \ldots Z_{j,a_j}| \leq |g_j\text{-}tree|(\sum_{r\ rhs\ in\ M} w(r, b_j) + \kappa b_j) + \sum_{\overline{\pi}\, z_{j,i}\ in\ \tau_{r_j}} |\overline{\pi}\, Z_{j,i}|$$

where $b_j = NL_j + A_M|g_j\text{-}tree|$. By *PC2.1* and since NL_j doesn't depend on critical arguments to f, we have $NL_j \le NL_f + A_M$, so

$$\begin{array}{rcl} b_j & = & NL_j + A_M|g_j\text{-}tree| \\ & \le & NL_f + A_M + A_M|g_j\text{-}tree| \\ & \le & NL_f + A_M + A_M|g_1\text{-}tree| + \cdots + A_M|g_k\text{-}tree| \\ & = & NL_f + A_M|f\text{-}tree| \\ & = & b \end{array}$$

so we have by Lemma 11 and induction hypotheses

$$\begin{array}{l} |f\,X_1 \ldots X_m| \\ = |r\sigma| \\ \le w(r, NL_f) + \sum_{\overline{\pi}\,x_i \text{ loose in } \tau_r} |\overline{\pi}\,X_i| + |g_1\,Z_{1,1} \ldots Z_{1,a_1}| + \cdots + |g_k\,Z_{k,1} \ldots Z_{k,a_k}| \\ \le w(r, NL_f) + (|g_1\text{-}tree| + \cdots + |g_k\text{-}tree|)(\sum w(r,b) + \kappa b) + \\ \sum_{\overline{\pi}\,x_i \text{ loose in } \tau_r} |\overline{\pi}\,X_i| + \sum_{\overline{\pi}\,z_{1,i} \text{ in } \tau_{r_1}} |\overline{\pi}\,Z_{1,i}| + \cdots + \sum_{\overline{\pi}\,z_{k,i} \text{ in } \tau_{r_k}} |\overline{\pi}\,Z_{k,i}| \end{array}$$

We wish to use *PC2.1* and Part 1 of this lemma to show that

$$\sum_{i \in cu_1} |Z_{1,i}| \le NL_f + \sum_{i \in cu_f \wedge \overline{\Pi}\,x_i \in \tau_{g_1 \overline{z}}} |\overline{\pi}\,X_i| + A_M \qquad (1)$$

(and the same for $2, \ldots, k$). *PC2.1* gives that

$$\sum_{i \in cu_1} |Z_{1,i}| \le NL_f + \sum_{i \in CU_f} |X_i| + C_M$$

whereas Part 1 of this lemma (this is why we introduced Part 1!) for $\tau_{g_1 \overline{z}}$ along with *LIC* for $\tau_{g_1 \overline{z}}$ yield that

$$\begin{array}{rcl} \sum_{i \in cu_1} |Z_{1,i}| & \le & \sum_{i \in cu_1} w(z_{1,i}, NL_f) + \sum_{\overline{\pi}\,x_j \in \tau_{z_{1,i}}} |\overline{\pi}\,X_j| \\ & \le & \sum_{i=1}^{a_1} w(z_{1,i}, NL_f) + \sum_{\overline{\pi}\,x_j \in \tau_{g_1 \overline{z}}} |\overline{\pi}\,X_j| \end{array}$$

The constant part of the polynomial $\sum_{i=1}^{a_1} w(z_i, NL_f)$ is 0 (by definition of w). So $\sum_{i \in cu_1} |Z_{1,i}|$ must be bounded by the sum of NL_f, $\sum_{\overline{\pi}\,x_j \in \tau_{g_1 \overline{z}}} |\overline{\pi}\,X_j|$, the number of critical positions in g_1, and C_M. So Equation 1 is ok.

Then we have by *LIC* for $\tau_{g_1}, \ldots, \tau_{g_k}$, by unit smallness

$$\begin{array}{l} \sum_{\overline{\pi}\,z_{1,i} \text{ in } \tau_{r_1}} |\overline{\pi}\,Z_{1,i}| + \cdots + \sum_{\overline{\pi}\,z_{k,i} \text{ in } \tau_{r_k}} |\overline{\pi}\,Z_{k,i}| \\ \le \sum_{i \in cu_1} |Z_{1,i}| + \cdots + \sum_{i \in cu_k} |Z_{1,k}| \\ \le NL_f + \sum_{i \in cu_f \wedge \overline{\Pi}\,x_i \in \tau_{g_1 \overline{z}}} |\overline{\pi}\,X_i| + A_M + \cdots + \\ NL_f + \sum_{i \in cu_f \wedge \overline{\Pi}\,x_i \in \tau_{g_k \overline{z}}} |\overline{\pi}\,X_i| + A_M \\ \le k(NL_f + A_M) + \sum_{\overline{\pi}\,x_i \in \tau_{g_1 \overline{z}}} |\overline{\pi}\,X_i| + \cdots + \sum_{\overline{\pi}\,x_i \in \tau_{g_k \overline{z}}} |\overline{\pi}\,X_i| \\ \le \kappa b + \sum_{\overline{\pi}\,x_i \in \tau_{g_1 \overline{z}}} |\overline{\pi}\,X_i| + \cdots + \sum_{\overline{\pi}\,x_i \in \tau_{g_k \overline{z}}} |\overline{\pi}\,X_i| \end{array}$$

So altogether by *LIC* for τ_r we have

$$\begin{array}{rcl} |f\,X_1 \ldots X_m| & \leq & w(r, NL_f) + \kappa b + (|g_1\text{-}tree| + \cdots + |g_k\text{-}tree|)(\sum w(r,b) + \kappa b) + \sum_{\overline{\pi}\,x_i \in \tau_r} |\overline{\pi}\,X_i| \\ & \leq & |f\text{-}tree|(\sum_{r\ rhs\ in\ M} w(r,b) + \kappa b) + \sum_{\overline{\pi}\,x_i \in \tau_r} |\overline{\pi}\,X_i| \end{array}$$

○

The following lemma (very similar to Lemma 9 and to Part 1 of lemma 10) is used in the proof of Lemma 10.

Lemma 11 (*DDC* output tree lemma) *Let a (possibly empty) terminating canonical system with a fit family $\mathcal{F}^{\mathcal{F}}$ and an output family $\mathcal{F}^{\mathcal{O}}$ be given such that $\mathcal{F}^{\mathcal{F}}$ is unit smaller than $\mathcal{F}^{\mathcal{O}}$ and such that* PBO' *holds for every M-set. Define a new M-set M with output and fit units, using the given system such that (at least)* LIC *holds.*

For any n-ary $f \in M$ with input $X_1, \ldots, X_n$, let $(f\,(c\,y_1 \ldots y_m)\,x_2 \ldots x_n) = r$ be the appropriate equation, let σ be the appropriate substitution, let τ_r be the needed output tree, let u be the needed output f-unit, let U be the needed fit f-unit. let $q_1, \ldots, q_k$ be the recursive call terms in τ_t such that in r they don't have any $h \in M$ above (outside) them, let $\mathrm{NL}_f = \sum_{i \in \mathrm{NU}} |X_i|$, then: For any subterm t in τ_r, except a proper subterm of $q_1, \ldots, q_k$ or a proper subterm of a maximal projection sequence $\overline{\pi}\,x_i$ where i is noncritical, we have

$$|t\sigma| \leq w(t, \mathrm{NL}_f) + \sum_{\overline{\pi}\,x_i \text{ loose in } \tau_t} |\overline{\pi}\,x_i\,\sigma| + |q_1\sigma| + \cdots + |q_k\sigma|$$

Proof of Lemma 11 We prove Lemma 11 by induction on the structure of t. If t is

- A noncritical variable: Ok.
- A loose $\overline{\pi}\,x_i$ or an "outermost" recursive call term: Ok.
- $(c\,t_1 \ldots t_m)$: By induction hypothesis for $t_j, 1 \leq j \leq m$ and by *LIC*

$$\begin{array}{rcl} |t\sigma| & \leq & 1 + |t_1\sigma| + \cdots + |t_m\sigma| \\ & \leq & 1 + w(t_1, NL_f) + \cdots + w(t_m, NL_f) + \sum_{\overline{\pi}\,x_i\ loose\ in\ \tau_t} |\overline{\pi}\,x_i\,\sigma| + |q_1\sigma| + \cdots + |q_k\sigma| \\ & = & w(c\,t_1 \ldots t_m, NL_f) + \sum_{\overline{\pi}\,x_i\ loose\ in\ \tau_t} |\overline{\pi}\,x_i\,\sigma| + |q_1\sigma| + \cdots + |q_k\sigma| \end{array}$$

- $(g\,t_1 \ldots t_m), g \notin M$

  PBO' holds for M_g, so there's a polynomial Q_{M_g} for M_g such that for the needed output g-unit v, where $v = nv \cup cv$, and for the needed fit g-unit V, where $V = NV \cup CV$

$$|(g\,t_1\dots t_m)\sigma| \le Q_{M_g}(\sum_{j\in NV}|t_j\sigma|) + \sum_{j\in cv}|t_j\sigma|$$

We have

- for $j \in NV$: By Lemma 12 (not by induction hypothesis), $|t_j\sigma| \le w(t_j, NL_f)$. So

$$\sum_{j\in NV}|t_j\sigma| \le \sum_{j\in NV} w(t_j, NL_f) \le \sum_{j\ noncrit} w(t_j, NL_f)$$

- for $j \in cv$ we have by induction hypothesis for t_j and by *LIC*

$$\sum_{j\in cv}|t_j\sigma| \le \sum_{i\ crit} w(t_i, NL_f) + \sum_{\overline{\pi}\,x_i\ loose\ in\ \tau_t} |\overline{\pi}\,x_i\,\sigma| + |q_1\sigma| + \dots + |q_k\sigma|$$

Altogether

$$\begin{array}{ll} |t\sigma| & \\ \le & Q_{M_g}(\sum_{j\in\ noncrit} w(t_j, NL_f)) + \sum_{i\ crit} w(t_i, NL_f) + \\ & \sum_{\overline{\pi}\,x_i\ loose\ in\ \tau_t} |\overline{\pi}\,x_i\,\sigma| + |q_1\sigma| + \dots + |q_k\sigma| \\ \le & w(t, NL_f) + \sum_{\overline{\pi}\,x_i\ loose\ in\ \tau_t} |\overline{\pi}\,x_i\,\sigma| + |q_1\sigma| + \dots + |q_k\sigma| \end{array}$$

○

The following lemma is used in the proof of Lemma 11 and Lemma 10.

Lemma 12 *Let a (possibly empty) terminating canonical system with a fit family $\mathcal{F}^{\mathcal{F}}$ and an output family $\mathcal{F}^{\mathcal{O}}$ be given such that $\mathcal{F}^{\mathcal{F}}$ is unit smaller than $\mathcal{F}^{\mathcal{O}}$ and such that* PBO' *holds for every M-set. Define a new M-set M using the given system*

Let f be in M with input $X_1, \dots, X_n$ let σ be the appropriate substiutition, let U be the needed fit unit. For any subterm t of the needed fit tree such that t doesn't contain any recursive call terms nor critical variables as subterms

$$|t\sigma| \le w(t, \mathrm{NL}_f)$$

where $\mathrm{NL}_f = \sum_{i\in\ \mathrm{NU}} |X_i|$.

Proof of Lemma 12 By induction on t. If t is

- a noncritical variable: ok

- $(c\,t_1 \ldots t_m)$: By induction hypothesis

$$\begin{array}{rcl} |t\sigma| & = & 1 + |t_1\sigma| + \cdots + |t_m\sigma| \\ & \leq & 1 + w(t_1, NL_f) + \cdots + w(t_m, NL_f) \\ & = & w(t, NL_f) \end{array}$$

- $(g\,t_1 \ldots t_m)$, $g \notin M$. By *PBO'* and by monotonicity of Q_{M_g}, by induction hypothesis

$$\begin{array}{rcl} |t\sigma| & \leq & Q_{M_g}(\sum_{j\ noncrit}(|t_j\sigma|) + \sum_{j\ crit}|t_j\sigma| \\ & \leq & Q_{M_g}(\sum_{j\ noncrit} w(t_j, NL_f)) + \sum_{j\ crit} w(t_j, NL_f) \\ & \leq & w(t, NL_f) \end{array}$$

$\bigcirc$