Monotone interpretations

Guillaume Bonfante

Monotone interpretations

Guillaume Bonfante

INRIA-LORIA B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex, France,

1 First order functional programming

1.1 Syntax of programs

A program is defined formally as a quadruple $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R} \rangle$ with $\mathcal{X}, \mathcal{F}$ and $\mathcal{C}$ three disjoint sets which represent respectively the variables, the function symbols and the constructors symbols, and $\mathcal{R}$ a finite set of rules defined below:

$$\begin{array}{lllll}
\texttt{(Constructor terms)} & \mathcal{T}(\mathcal{C}) \ni v & ::= & \mathbf{c} \mid \mathbf{c}(v_1, \cdots, v_n) \\
\texttt{(terms)} & \mathcal{T}(\mathcal{C}, \mathcal{F}, \mathcal{X}) \ni t & ::= & \mathbf{c} \mid x \mid \mathbf{c}(t_1, \cdots, t_n) \mid \mathtt{f}(t_1, \cdots, t_n) \\
\texttt{(patterns)} & \mathcal{P} \ni p & ::= & \mathbf{c} \mid x \mid \mathbf{c}(p_1, \cdots, p_n) \\
\texttt{(rules)} & \mathcal{R} \ni r & ::= & \mathtt{f}(p_1, \cdots, p_n) \to t
\end{array}$$

where $x \in \mathcal{X}$, $\mathtt{f} \in \mathcal{F}$, and $\mathbf{c} \in \mathcal{C}$.

The set of rules induces a rewriting relation $\to$. The relation $\xrightarrow{*}$ is the reflexive and transitive closure of $\to$. Throughout, we consider only orthogonal assemblies, that is, rule patterns are disjoint and linear. So each program is confluent [4].

The size $|t|$ of a term t defined to be the number of symbols of arity strictly greater than 0 occurring in it.

1.2 Semantics

The domain of computation of an program $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R} \rangle$ is the constructor algebra $\mathcal{T}(\mathcal{C})$. A substitution σ is a mapping from variables to terms and a ground substitution is one which ranges over constructor terms of $\mathcal{T}(\mathcal{C})$. Observe that constructor terms are normal forms for the program.

For each function symbol $\mathtt{f} \in \mathcal{F}$, the program $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R} \rangle$ computes a partial function $[\![\mathtt{f}]\!] : \mathcal{T}(\mathcal{C})^n \to \mathcal{T}(\mathcal{C})$ defined by: For all $v_i \in \mathcal{T}(\mathcal{C})$, $[\![\mathtt{f}]\!](v_1, \cdots, v_n) = w$ iff $\mathtt{f}(v_1, \cdots, v_n) \xrightarrow{*} w$ and w is in $\mathcal{T}(\mathcal{C})$. Otherwise $[\![\mathtt{f}]\!](v_1, \cdots, v_n)$ is undefined.

A term t which contains n variables $x_1, \cdots, x_n$ defines a function ϕ_t such that for all $v_i \in \mathcal{T}(\mathcal{C})$, $\phi_t(v_1, \cdots, v_n) = [\![t\sigma]\!]$ where σ is a ground substitution such that $x_i\sigma = v_i$ for any i.

2 Monotone interpretations

A monotone assignment of a symbol $b \in \mathcal{F} \bigcup \mathcal{C}$ of arity n is a weakely monotonic function $(\!|b|\!) : \mathbb{R}^n \to \mathbb{R}$, that is for all $X_1, \cdots, X_n, Y_1, \cdots, Y_n$ of $\mathbb{R}$ with $X_i \leq Y_i$, we have $(\!|b|\!)(X_1, \cdots, X_n) \leq (\!|b|\!)(Y_1, \cdots, Y_n)$.

We extend assignment $(\!|-|\!)$ to terms canonically. Given a term t with n variables, the assignment $(\!|t|\!)$ is a function $(\mathbb{R})^n \to \mathbb{R}$ defined by the rules:

$$(\!|b(t_1, \cdots, t_n)|\!) = (\!|b|\!)((\!|t_1|\!), \cdots, (\!|t_n|\!))$$
$$(\!|x|\!) = X$$

where X is a fresh variable ranging over reals.

Definition 1 (Monotone interpretations). *A monotone interpretation $(\!|-|\!)$ of a program $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R} \rangle$ is a monotone assignment such that for each rule $f(p_1, \ldots, p_k) \to r \in \mathcal{R}$, and for every constructor substitution σ*

$$(\!|f(p_1, \ldots, p_k)\sigma|\!) \geq (\!|r\sigma|\!)$$

and for all terms $t \lhd r\sigma$, either $(\!|t|\!) \leq (\!|p_i\sigma|\!)$ or $(\!|t|\!) \leq (\!|f(p_1, \ldots, p_k)\sigma|\!)$.

Notice that a quasi-interpretation is a particular case of monotone interpretation. See for instance [3]. Traditional polynomial interpretation are also monotone interpretations [2].

Definition 2. *An assignment $(\!|-|\!)$ is said to be polynomially bounded if for each symbol $b \in \mathcal{F} \bigcup \mathcal{C}$, $(\!|b|\!)$ is a function bounded by a polynomial. A monotone interpretation $(\!|-|\!)$ is polynomial if the assignment $(\!|-|\!)$ is polynomial.*

2.1 Additive assignments

Now, say that an assignment of a symbol b of arity $n > 0$ is additive if

$$(\!|b|\!)(X_1, \cdots, X_n) = \sum_{i=1}^{n} X_i + \alpha \text{ with } \alpha \geq 1$$

An assignment of a program $\mathtt{p}$ is additive if each constructor symbol of $\mathtt{p}$ has an additive assignment. *A program is additive if it admits a monotone interpretation which is an additive assignment.*

Example 1. Consider the program which computes the logarithm function and described by the following rules:

$$\begin{array}{r|r} \mathtt{log}(\mathbf{0}) \to \mathbf{0} & \mathtt{half}(\mathbf{0}) \to \mathbf{0} \\ \mathtt{log}(\mathbf{S}(\mathbf{0})) \to \mathbf{0} & \mathtt{half}(\mathbf{S}(\mathbf{0})) \to \mathbf{0} \\ \mathtt{log}(\mathbf{S}(\mathbf{S}(y))) \to \mathbf{S}(\mathtt{log}(\mathtt{half}(y))) & \mathtt{half}(\mathbf{S}(\mathbf{S}(y))) \to \mathbf{S}(\mathtt{half}(y)) \end{array}$$

It admits the following additive monotone interpretation:

$$(\!|\mathbf{0}|\!) = 0$$
$$(\!|\mathbf{S}|\!)(X) = X + 1$$
$$(\!|\mathtt{log}|\!)(X) = \log_2(X)$$
$$(\!|\mathtt{half}|\!)(X) = X/2$$

2.2 Key properties

Proposition 1. *Assume that $(\!|-|\!)$ is a monotone interpretation of a program $\mathtt{p}$. For any terms u and v such that $u \xrightarrow{*} v$, we have $(\!|u|\!) \geq (\!|v|\!)$*

Proof. A context is a particular term that we write $\mathsf{C}[\diamond]$ where $\diamond$ is a new variable. The substitution of $\diamond$ in $\mathsf{C}[\diamond]$ by a term t is noted $\mathsf{C}[t]$.

The proof goes by induction on the derivation length n. For this, suppose that $u = u_0 \to \ldots \to u_n = v$. If $n = 0$ then the result is immediate. Otherwise $n > 0$ and in this case, there is a rule $\mathtt{f}(p_1, \cdots, p_n) \to t$ and a constructor substitution σ such that $u_0 = \mathsf{C}[\mathtt{f}(p_1, \cdots, p_n)\sigma]$ and $u_1 = \mathsf{C}[t\sigma]$. Since $(\!|-|\!)$ is a monotone interpretation, we have $(\!|t\sigma|\!) \leq (\!|\mathtt{f}(p_1, \cdots, p_n)\sigma|\!)$. The monotonicity property implies that $(\!|\mathsf{C}[t\sigma]|\!) \leq (\!|\mathsf{C}[\mathtt{f}(p_1, \cdots, p_n)\sigma]|\!)$. We conclude by induction hypothesis.

Proposition 2. *Assume that $(\!|\_|\!)$ is an additive assignment.*

1. *For any constructor term t in $\mathcal{T}(\mathcal{C})$, we have $|t| \leq (\!|t|\!)$.*
2. *For any constructor term t in $\mathcal{T}(\mathcal{C})$, we have $(\!|t|\!) \leq k \times |t|$.*

Proof. The proof goes by induction on the size of t. □

Proposition 3. *Assume that $(\!|\_|\!)$ is an additive monotone interpretation of a program $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R} \rangle$. For any term u and any constructor term $t \in \mathcal{T}(\mathcal{C})$, if $u \xrightarrow{*} t$, we have $|t| \leq (\!|u|\!)$.*

Proof. It is a consequence of Proposition 1 and of the above Proposition 2. □

From the above results, we deduce a polynomial upper bound on the size of the values computed by function symbols of an additive program. In other words, monotone interpretations are a particular case of sup-interpretation. The motivation to use monotone interpretation vs sup-interpretation is that the synthesis of interpretations is decidable as quasi-interpretations are.

We consider the class Max-Poly of functions. It contains constant functions ranging over non-negative integers, closed by projections, maximum, addition, multiplication and composition.

Such functions can be written as $f = \max_{i \in I}(P_i)$ where I is finite and the P_i are polynomials. We call $\sharp(I)$ the max-degree of f and $\max_{i \in I}(\mathtt{deg}(P_i))$ the polynomial degree of f.

Theorem 1. *We suppose we are given some constants M and D. Then, given a program $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R} \rangle$, it is decidable whether there exists or not a monotone interpretation for $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R} \rangle$ such that all function symbol is interpreted by a function of max-degree bounded by M and polynomial degree bounded by D.*

The proof is essentially the one given in [3]. The only difference is that we have removed the monotonicity and we have introduced a property on the subterms of the right hand sides of the rules.

Lemma 1 (Fundamental Lemma). *Assume that $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R} \rangle$ is an additive program admitting a monotone interpretation $(\!|-|\!)$. Suppose we are given a function $\mathtt{f}$. Then, there is a polynomial Q such that for all constructor substitution σ,*

$$|[\![\mathtt{f}(x_1, \cdots, x_k)\sigma]\!]| \leq Q(\max_{i=1..n} |x_i\sigma|)$$

Proof. As $[\![\mathtt{f}(x_1, \cdots, x_k)\sigma]\!]$ is a constructor term,

$$\begin{aligned}
|[\![\mathtt{f}(x_1, \cdots, x_k)\sigma]\!]| &\leq (\!|[\![\mathtt{f}(x_1, \cdots, x_k)\sigma]\!]|\!) && \text{by Proposition 2(1)} \\
&\leq (\!|\mathtt{f}(x\sigma_1, \cdots, x\sigma_k)|\!) && \text{by Proposition 3} \\
&\leq P(\max_{i=1..n} (\!|x_i\sigma|\!)) && \text{since assigments are polynomially bounded} \\
&\leq P \circ k(\max_{i=1..n} |x_i\sigma|) && \text{by Proposition 2(2)}
\end{aligned}$$

3 Termination Ordering

Definition 3. *Given an ordering $>$, its lexicographic extension $>_L$ over sequences is defined by $(m_1, \cdots, m_k) >_L (n_1, \cdots, n_k)$ if and only if $\exists j \leq k : \forall i \leq j-1, m_i \geq n_i \wedge m_j > n_j$.*

Its product extension $>_M$ over sequences is defined by $(m_1, \cdots, m_k) >_M (n_1, \cdots, n_k)$ if and only if

(i) $\forall i \leq p, m_i \geq n_i$ *and*
(ii) $\exists j \leq k$ *such that* $m_j > n_j$.

Definition 4. *A definition $\mathtt{f}(p_1, \cdots, p_n) \to r$ induces a relation on function symbols. Say that $\mathtt{f}$ calls $\mathtt{g}$ if $\mathtt{g}$ appears in r. We note this relation $\to$. By reflexive-transitive closure, the relation induces a pre-order on function symbols, noted $\xrightarrow{*}$. The corresponding equivalence relation $\simeq$ is defined by $\mathtt{f} \simeq \mathtt{g} \Leftrightarrow (\mathtt{f} \xrightarrow{*} \mathtt{g} \wedge \mathtt{g} \xrightarrow{*} \mathtt{f})$. The strict partial corresponding order is noted $\prec$. We have $\mathtt{g} \prec \mathtt{f} \Leftrightarrow (\mathtt{f} \xrightarrow{*} \mathtt{g} \wedge \neg(\mathtt{f} \xrightarrow{*} \mathtt{g}))$.*

Definition 5. *A program has simple* dependency pairs *if for all rules $\mathtt{f}(p_1, \cdots, p_n) \to r$, for all $\mathtt{g}(t_1, \cdots, t_k) \trianglelefteq r$ such that $\mathtt{f} \simeq \mathtt{g}$, for all ground substitution σ, we have:*

$$((\!|p_1\sigma|\!), \ldots, (\!|p_n\sigma|\!)) >_L ((\!|t_1\sigma|\!), \ldots, (\!|t_k\sigma|\!))$$

where $>$ is the usual ordering.

Proposition 4. *A program which has simple dependency pairs terminates.*

Since simple dependency pairs are a particular case of dependency pairs, and, due to Aarts and Giesl [1], it terminates.

Theorem 2 (SPACE)**.** *The set of functions computed by additive programs which have simple dependency pairs compute exactly the set of functions computable in polynomial space.*

Definition 6. *A rule* $\mathtt{f}(p_1, \cdots, p_n) \to r$ *is* cons-free *if* r *contains no constructor symbols.*

Definition 7. *A function* $\mathtt{f}$ *is said to be a destructor if:*

1. *for all rule* $\mathtt{f}(p_1, \cdots, p_n) \to r$ *with* $t \triangleleft r$ *and* $\mathtt{g} \prec \mathtt{f}$, $\mathtt{g}$ *is a destructor,*
2. *for all rule* $\mathtt{f}(p_1, \cdots, p_n) \to r$ *is cons-free.*

By extension, a term is said to be a destructor if it composed only of destructor functions and variables.

Proposition 5. *If* $\mathtt{f}$ *is a destructor, then for all* $t_1, \ldots, t_k \in \mathcal{C}$*, we have* $[\![\mathtt{f}(t_1, \ldots, t_k)]\!] \trianglelefteq t_i$ *for some* $i \leq k$*. In other words, the output of a destructor is some subterm of some input.*

As an example of a destructor function, we propose to compute the half function.

$$
\begin{array}{rl}
\mathtt{pred}(x) = \textbf{case}\ \ x\ \textbf{of} & \\
\qquad\qquad \diamond \to \diamond & \\
\qquad\quad \mathbf{s}(x') \to x' & \\
\mathtt{incr}(x, y) = y - \mathtt{pred}(y - x) & \\
\mathtt{log}(x) = \textbf{case}\ \ x\ \textbf{of} & \\
\qquad\qquad \diamond \to \diamond & \\
\qquad\quad \mathbf{s}(x') \to \mathtt{incr}(\mathtt{log}(\mathtt{half}(x)), x) &
\end{array}
\qquad\qquad
\begin{array}{rl}
\mathtt{half}(x) = \textbf{case}\ \ x\ \textbf{of} & \\
\qquad\qquad \diamond \to \diamond & \\
\qquad\quad \mathbf{s}(\diamond) \to \diamond & \\
\mathbf{s}(\mathbf{s}(x')) \to \mathtt{incr}(\mathtt{half}(x), x) &
\end{array}
$$

Definition 8. *A program has structural recursive calls if for all rules* $\mathtt{f}(p_1, \cdots, p_n) \to r$*, for all* $\mathtt{g}(t_1, \cdots, t_k) \trianglelefteq r$ *such that* $\mathtt{f} \simeq \mathtt{g}$*,* t_i *is a destructive term.*

Theorem 3 (TIME)**.** *The set of functions computed by additive programs which have dependency pairs and structural recursive calls is exactly the set of functions computable in polynomial time.*

4 Proofs

It is clear that additive programs with structural recursive calls compute PTIME functions. Indeed, the simulation given in [3] use quasi-interpretations, and consequently monotone interpretations. Furthermore, due to the fact that they are ordered by PPO, the recursive calls fulfill the structural recursive call hypothesis. For PSPACE, an analogous discussion can be done that use LPO instead of PPO.

We consider now the other direction. The two main ingredients to say that a computation can be done within PSPACE are the following.

1. for any node of the call-tree [3], its size is polynomially bounded by the size of the input,
2. any branch of the call-tree has polynomial length (wrt the size of the input).

If moreover, recursive calls are done over sub-terms, then computations are in PTIME. Since this is the case of structural recursive calls, we only have to prove the two ingredients to get the proof of both theorems.

So that we need the following lemmas.

Definition 9. *A* state *is a tuple* $\langle \mathtt{h}, v_1, \cdots, v_p\rangle$ *where* $\mathtt{h}$ *is a function symbol of arity* p *and* $v_1, \ldots, v_p$ *are constructor terms. Assume that* $\eta_1 = \langle \mathtt{h}, v_1, \cdots, v_p\rangle$ *and* $\eta_2 = \langle \mathtt{g}, u_1, \cdots, u_m\rangle$ *are two states. A* transition *is a triplet* $\eta_1 \overset{e}{\rightsquigarrow} \eta_2$ *such that:*

(i) e *is a rule* $\mathtt{h}(p_1, \cdots, p_n) \to t$ *of* $\mathcal{R}$,
(ii) there is a substitution σ *such that* $p_i\sigma = v_i$ *for all* $1 \leq i \leq n$,
(iii) there is a subterm $\mathtt{g}(w_1, \cdots, w_m)$ *of* t *such that* $\sigma w_i \downarrow u_i$ *for all* $1 \leq i \leq m$.

$Transition(\mathtt{f})$ *is the set of all transitions between states.* $\overset{*}{\rightsquigarrow}$ *is the reflexive transitive closure of* $\cup_{e\in\mathcal{R}} \overset{e}{\rightsquigarrow}$.

Definition 10. *The* $\langle \mathtt{f}, t_1, \cdots, t_n\rangle$*-call graph is a rooted graph defined as follows: (i) nodes are states, (ii) the root is the state* $\langle \mathtt{f}, t_1, \cdots, t_n\rangle$*, (iii) for each state* η_1*, the children of* η_1 *are the set of states* $\{\eta_2 \mid \eta_1 \overset{e}{\rightsquigarrow} \eta_2 \in Transition(\mathtt{f})\}$.

Lemma 2. *Given a program* $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R}\rangle$*, the following hold:*

1. *if* $\eta_1 = \langle \mathtt{h}, v_1, \cdots, v_p\rangle \rightsquigarrow \eta_2 = \langle \mathtt{g}, u_1, \cdots, u_m\rangle$*. Then, for all* $i \leq k$*, either* $(\!|u_i|\!) \leq (\!|h(\boldsymbol{v})|\!)$ *or* $(\!|u_i|\!) \leq (\!|v_j|\!)$ *for some* j.
2. *if* $\eta_1 = \langle \mathtt{h}, v_1, \cdots, v_p\rangle \overset{*}{\rightsquigarrow} \eta_2 = \langle \mathtt{g}, u_1, \cdots, u_m\rangle$*. Then, for all* $i \leq k$*, either* $(\!|u_i|\!) \leq (\!|h(\boldsymbol{v})|\!)$ *or* $(\!|u_i|\!) \leq (\!|v_j|\!)$ *for some* j.

Proof. So, for (i), the situation is the following. We have $\mathtt{h}(v_1, \cdots, v_n) = \mathtt{h}(p_1, \cdots, p_n)\sigma \to C[\mathtt{g}(w_1, \cdots, w_m)\sigma]$ with σ a closed substitution and $w_i\sigma \overset{*}{\to} u_i$.

By definition of monotone interpretations, we have $(\!|w_i\sigma|\!) \leq (\!|\mathtt{h}(v_1, \cdots, v_n)|\!)$ or $(\!|w_i\sigma|\!) \leq (\!|p_j|\!)$ for some j. Due, to Lemma 1, we have $(\!|u_i|\!) \leq (\!|w_i\sigma|\!)$, and so (i) holds.

(ii) is obtained by induction on the length of the branch.

Lemma 3 (Call-graph node size). *Given a program* $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R}\rangle$*, there is a polynomial* P *such that for all* $t_1, \ldots, t_k \in \mathcal{T}(\mathcal{C})$*, for all call-tree node* $\langle \mathtt{g}, t'_1, \ldots, t'_m\rangle$ *we have* $\mathtt{size}(t'_j) \leq P(\mathtt{size}(\boldsymbol{t_i}))$.

Proof. It is a direct consequence of the previous Lemma together with Proposition 2. So, main ingredient (1) holds.

Lemma 4. *Suppose that a program* $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R}\rangle$ *has simple dependency pairs. Then, the following holds:*

1. *if* $\eta_1 = \langle \boldsymbol{h}, v_1, \cdots, v_p\rangle \rightsquigarrow \eta_2 = \langle \boldsymbol{g}, u_1, \cdots, u_m\rangle$ *with* $\boldsymbol{g} \simeq \boldsymbol{h}$. *Then, for all* $(\!|v_1, \cdots, v_n|\!) >_L (\!|u_1, \cdots, u_m|\!)$.
2. *if* $\eta_1 = \langle \boldsymbol{h}, v_1, \cdots, v_p\rangle \overset{*}{\rightsquigarrow} \eta_2 = \langle \boldsymbol{g}, u_1, \cdots, u_m\rangle$ *with* $\boldsymbol{g} \simeq \boldsymbol{h}$. *Then, for all* $(\!|v_1, \cdots, v_n|\!) >_L (\!|u_1, \cdots, u_m|\!)$.

Proof. As above, (ii) is a consequence of (i). So, we concentrate on the first statement. We have $\mathtt{h}(v_1, \cdots, v_n) = \mathtt{h}(p_1, \cdots, p_n)\sigma \to C[\mathtt{g}(w_1, \cdots, w_m)\sigma]$ with σ a closed substitution and $w_i\sigma \overset{*}{\to} u_i$. By simple dependency, we have $((\!|p_1\sigma|\!), \ldots, (\!|p_n\sigma|\!)) >_L ((\!|w_1\sigma|\!), \ldots, (\!|w_m\sigma|\!))$. Again, by Proposition 2, we have $((\!|p_1\sigma|\!), \ldots, (\!|p_n\sigma|\!)) >_L ((\!|u_1|\!), \ldots, (\!|u_m|\!))$.

Lemma 5. *Suppose that* $\boldsymbol{t^1} >_L \boldsymbol{t^2} >_L \ldots \boldsymbol{t^k}$ *where the vectors have length bounded by* d. *Suppose moreover that for any term* t_1 *appearing as a component of the latter vectors, there is a bound* A *on the length of any chain* $t_1 > t_2 > t_3 > \ldots$. *Then,* k *is bounded by* A^{d+1}.

The remaining of the section is to prove that we have the hypothesis of the Lemma. First, the following holds, as a consequence of Hilbert's Positivstellensatz.

Theorem 4 (Stengle 74). *Suppose that we are given polynomials* $P_1, \ldots, P_m \in \mathbb{R}[X_1, \ldots, X_k]$, *the two following points are equivalent:*

1. $\{X_1, \ldots, X_k : P_1(X_1, \ldots, X_k) \geq 0 \wedge \cdots \wedge P_m(X_1, \ldots, X_k) \geq 0\} = \emptyset$
2. $\exists Q_1, \ldots, Q_m : -1 = \sum_{i \leq m} Q_i P_i$ *where the* Q_i *are sum of squares (and so positive).*

Proposition 6. *Suppose that a program* $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R}\rangle$ *admits a monotone interpretation over Max-Poly. If* $(\!|t|\!)(X_1, \ldots, X_k) > (\!|u|\!)(X_1, \ldots, X_k)$ *for some terms* t *and* u, *then, there is a polynomial* $P_{t,u}(X_1, \ldots, X_k)$ *such that* $(\!|t|\!)(X_1, \ldots, X_k) - (\!|u|\!)(X_1, \ldots, X_k) \geq \dfrac{1}{P_{t,u}(X_1, \ldots, X_k)}$. *Observe that such a polynomial is strictly positive:* $\forall X_1, \ldots, X_k > 0 : P_{t,u}(X_1, \ldots, X_k) > 0$.

Proof. We can write $(\!|t|\!)(X_1, \ldots, X_k) = \max_{i \in I}(P_i(X_1, \ldots, X_k))$ and $(\!|u|\!)(X_1, \ldots, X_k) = \max_{j \in J}(Q_j(X_1, \ldots, X_k))$. So, for all j, we have $(\!|t|\!)(X_1, \ldots, X_k) > Q_j(X_1, \ldots, X_k)$. Suppose that we find for all these j a polynomial R_j such that

$$(\!|t|\!)(X_1, \ldots, X_k) \geq Q_j(X_1, \ldots, X_k) + \frac{1}{R_j(X_1, \ldots, X_k)}. \tag{1}$$

Then for all j, we have $(\!|t|\!)(X_1, \ldots, X_k) \geq Q_j(X_1, \ldots, X_k) + \dfrac{1}{\sum_{j \in J} R_j(X_1, \ldots, X_k)}$.

And so, $(\!|t|\!)(X_1, \ldots, X_k) \geq \max_{j \in J}(Q_j(X_1, \ldots, X_k)) + \dfrac{1}{\sum_{j \in J} R_j(X_1, \ldots, X_k)}$.

Consequently, we define $P_{t,u}(X_1, \ldots, X_k) = \sum_{j \in J} R_j(X_1, \ldots, X_k)$.

So, it remains to find the polynomials R_ℓ for all $\ell \in J$. Take j, one of them. We define D_i the set $\{(X_1, \ldots, X_k) \in (\mathbb{R}^+)^k : \forall j \neq i : P_i(X_1, \ldots, X_k) \geq$

$P_j(X_1,\dots,X_k)\}$. On D_i, $(\!|t|\!)(X_1,\dots,X_k) = P_i(X_1,\dots,X_k)$. And so, on D_i, we have $P_i(X_1,\dots,X_k) > Q_j(X_1,\dots,X_k)$.

As a consequence, for all $i \in I$, the set

$$\left\{\begin{array}{l} X_1 \geq 0,\dots,X_k \geq 0, \\ P_i(X_1,\dots,X_k) - P_1(X_1,\dots,X_k) \geq 0,\dots,P_i(X_1,\dots,X_k) - P_m(X_1,\dots,X_k) \geq 0, \\ Q_j(X_1,\dots,X_k) - P_i(X_1,\dots,X_k) \geq 0 \end{array}\right\}$$

is empty. And then, by Positivstellensatz, there are positive polynomials $T_{1,i},\dots,T_{k+m+1,i}$ such that

$$\begin{array}{l} -1 = X_1T_{1,i} + \cdots X_kT_{k,i} + \\ (P_i(X_1,\dots,X_k) - P_1(X_1,\dots,X_k))T_{k+1,i} + \cdots + (P_i(X_1,\dots,X_k) - P_m(X_1,\dots,X_k))T_{k+m,i} + \\ (Q_j(X_1,\dots,X_k) - P_i(X_1,\dots,X_k))T_{k+1+m,i}. \end{array}$$

Observe that on D_i, we have

$$(P_i(X_1,\dots,X_k) - Q_j(X_1,\dots,X_k))T_{k+1+m,i} \geq 1.$$

So that we define $R_j = \sum_{1\leq i\leq m} T_{k+1+m,i}$.

Proof (Ingredient 2). So, we take a branch of computation, that is a sequence $\eta_1 = \langle \mathtt{f}_1, v_{11},\cdots,v_{1p_1}\rangle \rightsquigarrow \eta_2\langle \mathtt{f}_2, v_{21},\cdots,v_{2p_2}\rangle \rightsquigarrow \cdots \rightsquigarrow \eta_k\langle \mathtt{f}_k, v_{k1},\cdots,v_{kp_k}\rangle$ of nodes.

Observe that it is compose of a first subsequence of nodes with function symbol equivalent to $\mathtt{f}_1$, then by an other subsequence of equivalent nodes smaller than $\mathtt{f}_1$, etc. Since the rank of a function is finite, we only need to consider the case where $\mathtt{f}_k$ is equivalent to $\mathtt{f}_1$.

In that case, Lemma 4 shows that $(\!|v_{11},\cdots,v_{1p_1}|\!) >_L (\!|v_{21},\cdots,v_{2p_2}|\!) > \cdots (\!|v_{k1},\cdots,v_{kp_k}|\!)$.

Now, take a chain of terms like in Lemma 5, by Lemma 3, we have $(\!|t_i|\!) \leq P(\max_i |x_i|)$ where the x_i are the inputs. By Proposition 6, we have for all i, $(\!|t_i|\!) > (\!|t_{i+1}|\!) + \dfrac{1}{Q((\!|t_i|\!),(\!|t_{i+1}|\!))} > (\!|t_{i+1}|\!) + \dfrac{1}{Q(P(\max_i |x_i|),P(\max_i |x_i|))}$. As a consequence, we have $k < Q(P(\max_i |x_i|),P(\max_i |x_i|)) \times ((\!|t_1|\!) - (\!|t_k|\!)) \leq Q(P(\max_i |x_i|),P(\max_i |x_i|)) \times (\!|t_1|\!)$ which is a polynomially bounded wrt the input.

As a consequence, since the arity of functions is bounded, we can apply Lemma 5 and we get a polynomial bound

$$[Q(P(\max_i |x_i|),P(\max_i |x_i|)) \times P(\max_i |x_i|),P(\max_i |x_i|)]^{d+1}$$

on the length of any branch of computations. Ingredient (2) holds.

References

1. T. Aarts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.

2. G. Bonfante, A. Cichon, J.-Y. Marion, and H. Touzet. Algorithms with polynomial interpretation termination proof. *Journal of Functional Programming*, 11(1):33–53, 2001.
3. G. Bonfante, J.-Y. Marion, and J.-Y. Moyen. Quasi-interpretation a way to control resources. *Accepted to Theoretical Computer Science*, 2007. `http://www.loria.fr/~marionjy`.
4. G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980.