Complexity by typing

Antoine Taveneaux directed by Frédéric Blanqui

Friday 17th July 2009

Table of contents

First introduction

This work is based on 2 line of work :

- Complexity in rewriting systems (by Jean-Yves Marion for example).
- Termination in rewriting systems using type size annotations (by Frédéric Blanqui for example)

Is it possible to extend the type system on the size to prove the termination of a function to a type system to bound the complexity ?

Question

How check indication about the complexity of a function with a annotations on types ?

First introduction

This work is based on 2 line of work :

- Complexity in rewriting systems (by Jean-Yves Marion for example).
- Termination in rewriting systems using type size annotations (by Frédéric Blanqui for example)

Is it possible to extend the type system on the size to prove the termination of a function to a type system to bound the complexity ?

Question

How check indication about the complexity of a function with a annotations on types ?

Table of contents :

Rewriting system

- We consider programmes define with a rewriting system and by first order terms.
- Terms are define on symbols of function, constructor and variable.

How dose it work ?

A rewriting system give rules to transform a term into another.

- When we cannot rewrite a term we say that it is in normal form.

An example

$(\mathcal{S} = \{0, s, +\}, \mathcal{C} = \{0, s\}, \mathcal{F} = \{+\}, \mathcal{E})$ with $\mathcal{E}$ describe by following equations :

$$\begin{cases} +\ 0\ y \rightarrow y \\ +\ s(x)\ y \rightarrow +\ x\ s(y) \end{cases}$$

On the term $+\ s(s(0))\ s(0)$:

$$+\ s(s(0))\ s(0) \rightarrow +\ s(0)\ s(s(0)) \rightarrow +\ 0\ s(s(s(0))) \rightarrow s(s(s(0)))$$

An example

$(\mathcal{S} = \{0, s, +\}, \mathcal{C} = \{0, s\}, \mathcal{F} = \{+\}, \mathcal{E})$ with $\mathcal{E}$ describe by following equations :

$$\begin{cases} +\ 0\ y \rightarrow y \\ +\ s(x)\ y \rightarrow +\ x\ s(y) \end{cases}$$

On the term $+\ s(s(0))\ s(0)$:

$+\ s(s(0))\ s(0) \rightarrow +\ s(0)\ s(s(0)) \rightarrow +\ 0\ s(s(s(0))) \rightarrow s(s(s(0)))$

An example

$(\mathcal{S} = \{0, s, +\}, \mathcal{C} = \{0, s\}, \mathcal{F} = \{+\}, \mathcal{E})$ with $\mathcal{E}$ describe by following equations :

$$\begin{cases} + \; 0 \; y \to y \\ + \; s(x) \; y \to + \; x \; s(y) \end{cases}$$

On the term $+ \; s(s(0)) \; s(0)$:

$$+ \; s(s(0)) \; s(0) \to + \; s(0) \; s(s(0)) \to + \; 0 \; s(s(s(0))) \to s(s(s(0)))$$

An example

$(\mathcal{S} = \{0, s, +\}, \mathcal{C} = \{0, s\}, \mathcal{F} = \{+\}, \mathcal{E})$ with $\mathcal{E}$ describe by following equations :

$$\begin{cases} +\ 0\ y \rightarrow y \\ +\ s(x)\ y \rightarrow +\ x\ s(y) \end{cases}$$

On the term $+\ s(s(0))\ s(0)$:

$$+\ s(s(0))\ s(0) \rightarrow +\ s(0)\ s(s(0)) \rightarrow +\ 0\ s(s(s(0))) \rightarrow s(s(s(0)))$$

Table of contents :

Complexity of calculus

The complexity of a derivation is the length of this derivation.

Definition

We say that a term $t \in \mathcal{T}(\mathcal{C} \cup \mathcal{F}, \mathcal{X})$ have a complexity $n \in \mathbb{N}$ if it is the length of the longest derivation :

$$\max\{n | \textit{there have a derivation with a length } n \textit{ beginning on } t\}$$

In other words, n is an upper bound for the length of all sequence of rewriting beginning on t.

Complexity of calculus

The complexity of a derivation is the length of this derivation.

Definition

We say that a term t $\in \mathcal{T}(\mathcal{C} \cup \mathcal{F}, \mathcal{X})$ have a complexity n $\in \mathbb{N}$ if it is the length of the longest derivation :

$$\max\{n | \textit{there have a derivation with a length n beginning on t}\}$$

In other words, n is an upper bound for the length of all sequence of rewriting beginning on t.

Complexity of calculus

The complexity of a derivation is the length of this derivation.

Definition

We say that a term t $\in \mathcal{T}(\mathcal{C} \cup \mathcal{F}, \mathcal{X})$ *have a complexity n* $\in \mathbb{N}$ *if it is the length of the longest derivation :*

$$\max\{n \mid \textit{there have a derivation with a length n beginning on t}\}$$

In other words, *n* is an upper bound for the length of all sequence of rewriting beginning on *t*.

Complexity of calculus

The complexity of a derivation is the length of this derivation.

Definition

We say that a term t $\in \mathcal{T}(\mathcal{C} \cup \mathcal{F}, \mathcal{X})$ *have a complexity n* $\in \mathbb{N}$ *if it is the length of the longest derivation :*

$$\max\{n \mid \text{there have a derivation with a length } n \text{ beginning on } t\}$$

In other words, n is an upper bound for the length of all sequence of rewriting beginning on t.

An example

For example

$$+\ s(s(0))\ s(0) \rightarrow +\ s(0)\ s(s(0)) \rightarrow +\ 0\ s(s(s(0))) \rightarrow s(s(s(0)))$$

have a complexity of 3 and it is the only one derivation. So the complexity of $+\ s(s(0))\ s(0)$ is 3.

Complexity of system

- We define the size of a term as the number of symbols used in this term. For example
- We define the complexity of a system as a function $c : \mathbb{N} \rightarrow \mathbb{N}$ such that for all $n \in \mathbb{N}$, $c(n)$ is the biggest complexity for a term with a size less than n.
- For the addition system we have $c(n) = n - 3$ for $n \geq 3$.

Complexity of system

- We define the size of a term as the number of symbols used in this term. For example
- We define the complexity of a system as a function $c : \mathbb{N} \rightarrow \mathbb{N}$ such that for all $n \in \mathbb{N}$, $c(n)$ is the biggest complexity for a term with a size less than n.
- For the addition system we have $c(n) = n - 3$ for $n \geq 3$.

Min or Max ?

- Some authors (Jean-Yves Marion for example) define the complexity with

  min$\{n|$there have a dérivation with a lenght n begining on $t\}$

- These definitions seem not compatible. For example with this system :

$$\begin{cases} + \; 0 \; y \rightarrow y \\ + \; s(x) \; y \rightarrow + \; x \; s(y) \\ f \; x \; y \rightarrow x \end{cases}$$

  And with the term :

$$f \; 0 \; (+ \; s(s(0)) \; s(0))$$

Min or Max ?

- Some authors (Jean-Yves Marion for example) define the complexity with

  $\min\{n | \text{there have a dérivation with a lenght } n \text{ begining on } t\}$

- These definitions seem not compatible. For example with this system :

$$\begin{cases} + \; 0 \; y \to y \\ + \; s(x) \; y \to + \; x \; s(y) \\ f \; x \; y \to x \end{cases}$$

And with the term :

$$f \; 0 \; (+ \; s(s(0)) \; s(0))$$

Another problem with these definitions

We have shown the following theorem :

Theorem

If a deterministic Turing machine can simulate a rewriting system (confluent on well formed input) with only a polynomial increasing of time on the complexity (defined by the Min) then :

$$P = NP \bigcap Co\text{-}NP$$

Idea of the proof

- For $\mathcal{L} \in \text{NP} \bigcap \text{Co-NP}$ there exist a polynomial algorithm $\mathcal{A}$ such that for all $x \in \mathcal{L}$ there exist a certificate c_x such that :

$$\mathcal{A}(x, c_x) = \begin{cases} 1 & \text{if } x \in \mathcal{L} \\ 0 & \text{if } x \notin \mathcal{L} \end{cases}$$

- In the execution we generate a random certificate and check it.
  - If the certificate is correct stops on output.
  - else we enumerate all certificates to find a correct certificate.
- The length of the shortest execution is bounded by polynomial.

Hot spot in the proof

- How transform a Turing machine into a rewriting system ?
- How to do represent a sequence of calculi by rewriting system.

Hot spot in the proof

- How transform a Turing machine into a rewriting system ?
- How to do represent a sequence of calculi by rewriting system.

Min and Max in a simple case

Proposition

In a rewriting system without critical pairs all innermost reductions have the same size.

In this case the distinction with Min and Max is not a problem.

Min and Max in a simple case

Proposition

In a rewriting system without critical pairs all innermost reductions have the same size.

In this case the distinction with Min and Max is not a problem.

Table of contents :

Size bounding for termination

- F. Blanqui (and others) use annotation system on the types to bound the output size.
- This system is powerful to show the termination of some functions.
- We want re-use this system to provide a new annotation system to bound the complexity.

Size bounding for termination

- F. Blanqui (and others) use annotation system on the types to bound the output size.
- This system is powerful to show the termination of some functions.
- We want re-use this system to provide a new annotation system to bound the complexity.

Size bounding for termination

- F. Blanqui (and others) use annotation system on the types to bound the output size.
- This system is powerful to show the termination of some functions.
- We want re-use this system to provide a new annotation system to bound the complexity.

Bound for complexity

- J.Y. Marion (and others) provide tools to study the complexity of some rewriting systems.
- With these bounds Marion can characterize polynomial time bounded functions with a set of rewriting system.
- The complexity bounds are relay huge (but polynomial). In most cases these bounds are reasonable in comparison of the real complexity.
- In Marion's work the link with complexity and size is relay important.

Bound for complexity

- J.Y. Marion (and others) provide tools to study the complexity of some rewriting systems.
- With these bounds Marion can characterize polynomial time bounded functions with a set of rewriting system.
- The complexity bounds are relay huge (but polynomial). In most cases these bounds are reasonable in comparison of the real complexity.
- In Marion's work the link with complexity and size is relay important.

Bound for complexity

- J.Y. Marion (and others) provide tools to study the complexity of some rewriting systems.
- With these bounds Marion can characterize polynomial time bounded functions with a set of rewriting system.
- The complexity bounds are relay huge (but polynomial). In most cases these bounds are reasonable in comparison of the real complexity.
- In Marion's work the link with complexity and size is relay important.

Bound for complexity

- J.Y. Marion (and others) provide tools to study the complexity of some rewriting systems.
- With these bounds Marion can characterize polynomial time bounded functions with a set of rewriting system.
- The complexity bounds are relay huge (but polynomial). In most cases these bounds are reasonable in comparison of the real complexity.
- In Marion's work the link with complexity and size is relay important.

Table of contents :

Recursive primitive functions model

Definition

- *two constructors :* $0 : Nat$ *and* $Succ : Nat \rightarrow Nat$.
- *Projections :* $p_i(x_1, ..., x_k) \rightarrow x_i$

And with these constructions :

- *Composition : with* $g_1, g_2, \ldots, g_k$ *and h primitive recursive functions with good arity then the function* $f = h(g_1, ..., g_k)$ *is an primitive recursive function.*
- *Recursive definition : if g have an arity n, and h* $n+2$*, we define a new recursive primitive function* $\mu_{g,h}$ *:*

$$\begin{cases} \mu_{g,h}(0, \vec{y}) = g(\vec{y}) \\ \mu_{g,h}(Succ(x), \vec{y}) = h(x, \mu_{g,h}(x, \vec{y}), \vec{y}) \end{cases}$$

A Light μ

The original μ is :

$$\begin{cases} \mu_{g,h}(0, \vec{y}) = g(\vec{y}) \\ \mu_{g,h}(Succ(x), \vec{y}) = h(x, \mu_{g,h}(x, \vec{y}), \vec{y}) \end{cases}$$

In some cases we can consider a simpler version of μ :

$$\begin{cases} \mu'_{g,h}(0, \vec{y}) = g(\vec{y}) \\ \mu'_{g,h}(Succ(x), \vec{y}) = h(\mu_{g,h}(x, \vec{y})) \end{cases}$$

Innermost

- We are interested in the innermost strategy.
- This strategy allow to look only the size of normal term.
- In the recursive primitive functions model a bound for the innermost is a bound for all reductions.

Innermost

- We are interested in the innermost strategy.
- This strategy allow to look only the size of normal term.
- In the recursive primitive functions model a bound for the innermost is a bound for all reductions.

Innermost

- We are interested in the innermost strategy.
- This strategy allow to look only the size of normal term.
- In the recursive primitive functions model a bound for the innermost is a bound for all reductions.

Bound the size for the primitive recursive functions (*Type*, *size*)

$$\frac{}{0 : (Nat; 1)} \qquad \frac{}{Succ : (Nat \rightarrow Nat; n \rightarrow n + 1)}$$

$$\frac{}{p_i : (Nat \rightarrow \cdots \rightarrow Nat \rightarrow Nat; \alpha_1 \rightarrow \alpha_2 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \alpha_i)}$$

$$\frac{g : (Nat \rightarrow Nat; n \rightarrow n + k_g) \quad h : (Nat \rightarrow Nat; n \rightarrow n + k_h)}{g \circ h : (Nat \rightarrow Nat; n \rightarrow n + k_g + k_h)}$$

$$\frac{g : (Nat \rightarrow Nat; n \rightarrow n + k_g) \quad h : (Nat \rightarrow Nat; n \rightarrow n + k_h)}{\mu_{g,h} : (Nat \rightarrow Nat \rightarrow Nat; n \rightarrow m \rightarrow n(n + m + k_h) + k_g)}$$

Bound the size for the primitive recursive functions $(\mathit{Type}, \mathit{size})$

$$\frac{}{0 : (Nat; 1)} \qquad \frac{}{Succ : (Nat \rightarrow Nat; n \rightarrow n+1)}$$

$$\frac{}{p_i : (Nat \rightarrow \cdots \rightarrow Nat \rightarrow Nat; \alpha_1 \rightarrow \alpha_2 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \alpha_i)}$$

$$\frac{g : (Nat \rightarrow Nat; n \rightarrow n + k_g) \quad h : (Nat \rightarrow Nat; n \rightarrow n + k_h)}{g \circ h : (Nat \rightarrow Nat; n \rightarrow n + k_g + k_h)}$$

$$\frac{g : (Nat \rightarrow Nat; n \rightarrow n + k_g) \quad h : (Nat \rightarrow Nat; n \rightarrow n + k_h)}{\mu_{g,h} : (Nat \rightarrow Nat \rightarrow Nat; n \rightarrow m \rightarrow n(n + m + k_h) + k_g)}$$

Bound the size for the primitive recursive functions (*Type*, *size*)

$$\frac{}{0 : (Nat; 1)} \qquad \frac{}{Succ : (Nat \to Nat; n \to n+1)}$$

$$\frac{}{p_i : (Nat \to \cdots \to Nat \to Nat; \alpha_1 \to \alpha_2 \to \cdots \to \alpha_n \to \alpha_i)}$$

$$\frac{g : (Nat \to Nat; n \to n + k_g) \quad h : (Nat \to Nat; n \to n + k_h)}{g \circ h : (Nat \to Nat; n \to n + k_g + k_h)}$$

$$\frac{g : (Nat \to Nat; n \to n + k_g) \quad h : (Nat \to Nat; n \to n + k_h)}{\mu_{g,h} : (Nat \to Nat \to Nat; n \to m \to n(n + m + k_h) + k_g)}$$

Bound the size for the primitive recursive functions (*Type*, *size*)

$$\frac{}{0 : (Nat; 1)} \qquad \frac{}{Succ : (Nat \to Nat; n \to n + 1)}$$

$$\frac{}{p_i : (Nat \to \cdots \to Nat \to Nat; \alpha_1 \to \alpha_2 \to \cdots \to \alpha_n \to \alpha_i)}$$

$$\frac{g : (Nat \to Nat; n \to n + k_g) \quad h : (Nat \to Nat; n \to n + k_h)}{g \circ h : (Nat \to Nat; n \to n + k_g + k_h)}$$

$$\frac{g : (Nat \to Nat; n \to n + k_g) \quad h : (Nat \to Nat; n \to n + k_h)}{\mu_{g,h} : (Nat \to Nat \to Nat; n \to m \to n(n + m + k_h) + k_g)}$$

Bound the size for the primitive recursive functions $Type_{size}$

$$\frac{}{0 : Nat_1} \qquad \frac{}{Succ : Nat_n \to Nat_{n+1}}$$

$$\frac{}{p_i : Nat_{\alpha_1} \to Nat_{\alpha_2} \to \cdots \to Nat_{\alpha_n} \to Nat_{\alpha_i}}$$

$$\frac{g : Nat_n \to Nat_{n+k_g} \quad h : Nat_n \to Nat_{n+k_h}}{g \circ h : Nat_n \to Nat_{n+k_g+k_h}}$$

$$\frac{g : Nat_n \to Nat_{n+k_g} \quad h : Nat_n \to Nat_{n+k_h}}{\mu_{g,h} : Nat_n \to Nat_m \to Nat_{n(n+m+k_h)+k_g}}$$

From the size to the complexity : ($Type_{size}$, *complexity*)

The complexity is defined by a function (in $\mathbb{N}^k \to \mathbb{N}$) such that if f has a complexity $\mathcal{C}_f$ then all term ft with t in normal form and with a size less than n have a complexity $\mathcal{C}_f(n)$.

$$\frac{}{0 : (Nat_1\,;\, n \mapsto 0)} \qquad \frac{}{Succ : (Nat_n \to Nat_{n+1}\,;\, n \mapsto 0)}$$

$$\frac{}{p_i : (Nat_{\alpha_1} \to \cdots \to Nat_{\alpha_n} \to Nat_{\alpha_i}\,;\, n_1, \ldots, n_n \mapsto 1)}$$

$$\frac{g : (Nat_n \to Nat_{n+k_g}\,;\, \mathcal{C}_g) \quad h : (Nat_n \to Nat_{n+k_h}\,;\, \mathcal{C}_h)}{g \circ h : (Nat_n \to Nat_{n+k_g+k_h}\,;\, n \mapsto \mathcal{C}_h(n) + \mathcal{C}_g(n + k_h))}$$

From the size to the complexity : ($Type_{size}$, *complexity*)

The complexity is defined by a function (in $\mathbb{N}^k \to \mathbb{N}$) such that if f has a complexity $\mathcal{C}_f$ then all term ft with t in normal form and with a size less than n have a complexity $\mathcal{C}_f(n)$.

$$\frac{}{0 : (Nat_1 ; n \mapsto 0)} \qquad \frac{}{Succ : (Nat_n \to Nat_{n+1} ; n \mapsto 0)}$$

$$\frac{}{p_i : (Nat_{\alpha_1} \to \cdots \to Nat_{\alpha_n} \to Nat_{\alpha_i} ; n_1, \ldots, n_n \mapsto 1)}$$

$$\frac{g : (Nat_n \to Nat_{n+k_g} ; \mathcal{C}_g) \quad h : (Nat_n \to Nat_{n+k_h} ; \mathcal{C}_h)}{g \circ h : (Nat_n \to Nat_{n+k_g+k_h} ; n \mapsto \mathcal{C}_h(n) + \mathcal{C}_g(n + k_h))}$$

From the size to the complexity : ($Type_{size}$, *complexity*)

The complexity is defined by a function (in $\mathbb{N}^k \to \mathbb{N}$) such that if f has a complexity $\mathcal{C}_f$ then all term ft with t in normal form and with a size less than n have a complexity $\mathcal{C}_f(n)$.

$$\frac{}{0 : (Nat_1 ; n \mapsto 0)} \qquad \frac{}{Succ : (Nat_n \to Nat_{n+1} ; n \mapsto 0)}$$

$$\frac{}{p_i : (Nat_{\alpha_1} \to \cdots \to Nat_{\alpha_n} \to Nat_{\alpha_i} ; n_1, \ldots, n_n \mapsto 1)}$$

$$\frac{g : (Nat_n \to Nat_{n+k_g} ; \mathcal{C}_g) \quad h : (Nat_n \to Nat_{n+k_h} ; \mathcal{C}_h)}{g \circ h : (Nat_n \to Nat_{n+k_g+k_h} ; n \mapsto \mathcal{C}_h(n) + \mathcal{C}_g(n + k_h))}$$

From the size to the complexity : ($Type_{size}$, $complexity$)

$$\frac{g : (Nat_n \rightarrow Nat_{n+k_g}; \mathcal{C}_g) \quad h : (Nat_n \rightarrow Nat_{n+k_h}; \mathcal{C}_h)}{\mu_{g,h} : (Nat_n \rightarrow Nat_m \rightarrow Nat_{n(n+m+k_h)+k_g}; (n, m) \mapsto t)}$$

With

$$t = \mathcal{C}_g(m) + \sum_{x=1}^{n} \mathcal{C}_h(x + x(x + m + k_h) + k_g + m)$$

An example

When $C_g(n) \leq n^\alpha$ and $C_h(n) \leq n^\beta$ then :

$$\begin{aligned} t &= C_g(m) + \sum_{x=1}^{n} C_h(x + x(x + m + k_h) + k_g + m) \\ &\leq m^\alpha + \sum_{x=1}^{n} (x + x(x + m + k_h) + k_g + m)^\beta \\ &\leq m^\alpha + \sum_{x=1}^{n} x^{2\beta}(1 + k_g + k_h + 2m) \\ &\leq m^\alpha + n^{2\beta+1}(1 + k_g + k_h + 2m) \end{aligned}$$

An example

When $C_g(n) \leq n^\alpha$ and $C_h(n) \leq n^\beta$ then :

$$
\begin{aligned}
t &= \mathcal{C}_g(m) + \sum_{x=1}^{n} \mathcal{C}_h(x + x(x + m + k_h) + k_g + m) \\
&\leq m^\alpha + \sum_{x=1}^{n} (x + x(x + m + k_h) + k_g + m)^\beta \\
&\leq m^\alpha + \sum_{x=1}^{n} x^{2\beta}(1 + k_g + k_h + 2m) \\
&\leq m^\alpha + n^{2\beta+1}(1 + k_g + k_h + 2m)
\end{aligned}
$$

Sum

We can define the sum of 2 integer with :

$$\begin{cases} h(x_1, x_2, x_3) = Succ(p_2(x_1, x_2, x_3)) \\ g(y) = y \\ Sum(x, y) = \mu_{g,h}(x, y) \end{cases}$$

- With $|h(x_1, x_2, x_3)| \leq 1 + |x_2|$ and $|g(y)| = |y|$.
- So : $Sum(x, y)$ have a size bounded by $|x|(|x| + |y| + 1)$.

Sum

We can define the sum of 2 integer with :

$$\begin{cases} h(x_1, x_2, x_3) = Succ(p_2(x_1, x_2, x_3)) \\ g(y) = y \\ Sum(x, y) = \mu_{g,h}(x, y) \end{cases}$$

- With $|h(x_1, x_2, x_3)| \leq 1 + |x_2|$ and $|g(y)| = |y|$.
- So : $Sum(x, y)$ have a size bounded by $|x|(|x| + |y| + 1)$.

Sum

We can define the sum of 2 integer with :

$$\begin{cases} h(x_1, x_2, x_3) = Succ(p_2(x_1, x_2, x_3)) \\ g(y) = y \\ Sum(x, y) = \mu_{g,h}(x, y) \end{cases}$$

- With $|h(x_1, x_2, x_3)| \leq 1 + |x_2|$ and $|g(y)| = |y|$.
- So : $Sum(x, y)$ have a size bounded by $|x|(|x| + |y| + 1)$.

Problem

So, with the classical μ we cannot bound the complexity of the multiplication.

But with the μ' we can :

$$\begin{cases} \mu'_{g,h}(0, \vec{y}) = g(\vec{y}) \\ \mu'_{g,h}(Succ(x), \vec{y}) = h(\mu_{g,h}(x, \vec{y})) \end{cases}$$

Problem

So, with the classical μ we cannot bound the complexity of the multiplication.

But with the μ' we can :

$$\begin{cases} \mu'_{g,h}(0,\vec{y}) = g(\vec{y}) \\ \mu'_{g,h}(Succ(x),\vec{y}) = h(\mu_{g,h}(x,\vec{y})) \end{cases}$$

Summary of the work

- We have study the relation with the complexity defined by Min and Max.
- The link with the Turing complexity and the complexity in rewriting system.
- We have provide a bound for the complexity in the recursive primitives function model based on bounds of the output size.

Summary of the work

- We have study the relation with the complexity defined by Min and Max.
- The link with the Turing complexity and the complexity in rewriting system.
- We have provide a bound for the complexity in the recursive primitives function model based on bounds of the output size.

Summary of the work

- We have study the relation with the complexity defined by Min and Max.
- The link with the Turing complexity and the complexity in rewriting system.
- We have provide a bound for the complexity in the recursive primitives function model based on bounds of the output size.

Summary of the work

- We have study the relation with the complexity defined by Min and Max.
- The link with the Turing complexity and the complexity in rewriting system.
- We have provide a bound for the complexity in the recursive primitives function model based on bounds of the output size.

Future work

- Re-use a more formal annotation system (for example the Blanqui's one).
- Generalisation for a true type system (and not only for first order).
- Generalisation for a more general set of rewriting system.
- How check the complexity of a set of function ? How infer it ?

Future work

- Re-use a more formal annotation system (for example the Blanqui's one).
- Generalisation for a true type system (and not only for first order).
- Generalisation for a more general set of rewriting system.
- How check the complexity of a set of function ? How infer it ?

Future work

- Re-use a more formal annotation system (for example the Blanqui's one).
- Generalisation for a true type system (and not only for first order).
- Generalisation for a more general set of rewriting system.
- How check the complexity of a set of function ? How infer it ?

Future work

- Re-use a more formal annotation system (for example the Blanqui's one).
- Generalisation for a true type system (and not only for first order).
- Generalisation for a more general set of rewriting system.
- How check the complexity of a set of function ? How infer it ?

Problem about these generalisation

- For the moment we are bounded by *PTIME* function (in base 1), and a more powerful system of annotation should be extend this field.

Conclusion

- A link with time complexity and space complexity seems is clear.
- It is not easy to propose good restriction to provide an interesting bounds.
- An interesting field.

Conclusion

- A link with time complexity and space complexity seems is clear.
- It is not easy to propose good restriction to provide an interesting bounds.
- An interesting field.

Conclusion

- A link with time complexity and space complexity seems is clear.
- It is not easy to propose good restriction to provide an interesting bounds.
- An interesting field.

Thank you !

Questions ?

Thank you !

Questions ?