Estimation of the length of interactions in arena game semantics

Pierre Clairambault
University of Bath
p.clairambault@bath.ac.uk

Abstract

We estimate the maximal length of interactions between strategies in HO/N game semantics, in the spirit of the work by Schwichtenberg and Beckmann for the length of reduction in simply typed λ-calculus. Because of the operational content of game semantics, the bounds presented here also apply to head linear reduction on λ-terms and to the execution of programs by abstract machines (PAM/KAM), including in presence of computational effects such as non-determinism or ground type references. The proof proceeds by extracting from the games model a combinatorial rewriting rule on trees of natural numbers, which can then be analysed independently of game semantics or λ-calculus.

1 Introduction

Among the numerous notions of execution that one can consider on higher-order programming languages (in particular on the λ-calculus) *head linear reduction* [9] plays a particular role. Although it is not as widespread and specifically studied as, say, β-reduction, it is nonetheless implicit to various approaches of higher-order computation, such as geometry of interaction, game semantics, optimal reduction and ordinary operational semantics. It is also implicit to several abstract machines, including the Krivine Abstract Machine (KAM) [17] and the Pointer Abstract Machine (PAM) [9], in the sense that it is the reduction they perform [9, 5] and as such is a valuable abstraction of how programs are executed in the implementation of higher order languages.

Despite being closer to the implementation of programming languages, head linear reduction never drew a lot of attention from the community. Part of the reason for that is that it is not a usual notion of reduction: defining it properly on λ-terms requires both to extend the notion of redex and to restrict to linear substitution, leading to rather subtle and tricky definitions which lack the canonicity of β-reduction[1]. Moreover, its associated observational equivalence

[1] However, there are syntaxes on which head linear reductions appear more canonical that β-reduction, for instance proof nets [21].

is the same as for the usual head β-reduction, which makes it non relevant as long as one is interested in the equational theory of λ-calculus. However, head linear reduction should appear in the foreground as soon as one is interested in quantitative aspects of computation, such as complexity. On the contrary, although very precise bounds are known for the possible length of β-reduction chains in simply typed λ-calculus [24, 3], to the author's knowledge, the situation for head linear reduction remains essentially unexplored. Even if it is generally expected that the bounds remain hyper-exponential[2] (and this indeed what we will prove), it does not seem to follow easily from the bounds known for β-reduction.

Rather than reasoning directly on head linear reduction, we will instead look at it through game semantics [16]. Indeed, there is a close relationship between head linear reduction and interaction in games model of programming languages [8]. More precisely, given two β-normal and η-long λ-terms S and T, there is a step-by-step correspondence between head linear reduction chains of ST and game-theoretic interactions between the strategies $[\![S]\!]$ and $[\![T]\!]$. Of course, game semantics are not central to our analysis: as is often the case, our methods and results could be adapted to a purely syntactical framework. However, games have this considerable advantage of accommodating in a single framework purely functional programming languages such as the λ-calculus or PCF and a number of computational features such as non-determinism [15], control operators [19] and references [1]. This will allow us to do our study with an increased generality: our complexity results will hold for a variety of settings, from simply typed λ-calculus to richer languages possibly featuring the computational effects mentioned above, as long as there is no fixed point operator.

Outline. In Section 2 we will recall some of the basic definitions of Hyland-Ong game semantics, define the central notion of size of a strategy, and introduce our main question as the problem of finding the maximal length of an interaction between two strategies of fixed size. Our approach will be then to progressively simplify this problem in order to reach its underlying combinatorial nature. In Section 3 we first introduce the notion of *visible pointer structures*, *i.e.* plays where the identity of moves has been forgotten. This allows a more elementary (strategy-free) equivalent statement of our problem. Then we show how each position in a visible pointer structure can be characterised by a tree of natural numbers called an *agent*. We then show that the problem can be once again reformulated as the maximal length of a reduction on these agents. In Section 4 we study the length of this reduction, giving in particular an upper bound. We also give a corresponding lower bound, and finally use our result to estimate the maximal length of head linear reduction sequences on simply typed λ-terms.

[2] Note however that in [10], De Bruijn gives an upper bound for his local β-reduction, akin to head linear reduction. The bound is an iterate of the diagonal of an Ackermann-like function!

Related works. Our results and part of our methods are similar to the works of Schwichtenberg and Beckmann [24, 3], but the reduction we study is in some sense more challenging, because redexes are not destroyed as they are reduced. Moreover, the game semantics setting allows for an extra generality. The present work also has common points with work by Dal Lago and Laurent [18], in the sense that it uses tools from game semantics to reason on the length of execution. However the approach is very different : their estimate is very precise but uses an information on terms difficult to compute (almost as hard as actually performing execution). Here, we need little information on terms (gathering this information is linear in the size of the term), but our bounds are, in most cases, very rough.

2 Arena game semantics

We recall briefly the now usual definitions of arena games, first introduced in [16]. More detailed accounts can be found in [22, 13]. We are interested in games with two participants: Opponent (O, the *environment*) and Player (P, the *program*).

2.1 Arenas and Plays

Valid plays are generated by directed graphs called *arenas*, which are semantic versions of *types*. Formally, an **arena** is a structure $A = (M_A, \lambda_A, I_A, \vdash_A)$ where:

- M_A is a set of **moves**,
- $\lambda_A : M_A \to \{O, P\}$ is a polarity function indicating whether a move is an Opponent or Player move (O-move or P-move).
- $I_A \subseteq \lambda_A^{-1}(\{O\})$ is a set of **initial moves**.
- $\vdash_A \subset M_A \times M_A$ is a relation called **enabling**, such that if $m \vdash_A n$, then $\lambda_A(m) \neq \lambda_A(n)$.

In other words, an arena is just a directed bipartite graph. We now define plays as **justified sequences** over A: these are sequences s of moves of A, each non-initial move m in s being equipped with a pointer to an earlier move n in s, satisfying $n \vdash_A m$. In other words, a justified sequence s over A is such that each reversed pointer chain $s_{i_0} \leftarrow s_{i_1} \leftarrow \dots \leftarrow s_{i_n}$ is a path on A (viewed as a graph). The role of pointers is to allow *reopenings* or *backtracking* in plays. When writing justified sequences, we will often omit the justification information if this does not cause any ambiguity. The symbol $\sqsubseteq$ will denote the prefix ordering on justified sequences, and $s_1 \sqsubseteq^P s_2$ will mean that s_1 is a P-ending prefix of s_2. If s is a justified sequence on A, $|s|$ will denote its length.

Given a justified sequence s on A, it has two subsequences of particular interest: the P-view and O-view. The view for P (resp. O) may be understood as the subsequence of the play where P (resp. O) only sees his own duplications.

Practically, the P-view $\ulcorner s \urcorner$ of s is computed by forgetting everything under Opponent's pointers, in the following recursive way:

- $\ulcorner sm \urcorner = \ulcorner s \urcorner m$ if $\lambda_A(m) = P$;
- $\ulcorner sm \urcorner = m$ if $m \in I_A$ and m has no justification pointer;
- $\ulcorner s_1 m s_2 n \urcorner = \ulcorner s \urcorner mn$ if $\lambda_A(n) = O$ and n points to m.

The O-view $\llcorner s \lrcorner$ of s is defined dually, without the special treatment of initial moves[3]. The *legal plays* over A, denoted by $\mathcal{L}_A$, are the justified sequences s on A satisfying the **alternation** condition, *i.e.* that if $tmn \sqsubseteq s$, then $\lambda_A(m) \neq \lambda_A(n)$.

2.2 Classes of strategies

In this subsection, we will present several classes of strategies on arena games that are of interest to us in the present paper. A **strategy** σ on A is a set of even-length legal plays on A, closed under even-length prefix. A strategy from A to B is a strategy $\sigma : A \Rightarrow B$, where $A \Rightarrow B$ is the usual arrow arena defined by $M_{A\Rightarrow B} = M_A + M_B$, $\lambda_{A\Rightarrow B} = [\overline{\lambda_A}, \lambda_B]$ (where $\overline{\lambda_A}$ means λ_A with polarity O/P reversed), $I_{A\Rightarrow B} = I_B$ and $\vdash_{A\Rightarrow B} = \vdash_A + \vdash_B + I_B \times I_A$.

Composition. We define composition of strategies by the usual parallel interaction plus hiding mechanism. If A, B and C are arenas, we define the set of **interactions** $I(A, B, C)$ as the set of justified sequences u over A, B and C such that $u_{\restriction_{A,B}} \in \mathcal{L}_{A\Rightarrow B}$, $u_{\restriction_{B,C}} \in \mathcal{L}_{B\Rightarrow C}$ and $u_{\restriction_{A,C}} \in \mathcal{L}_{A\Rightarrow C}$. Then, if $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$, we define parallel interaction as $\sigma \parallel \tau = \{u \in I(A, B, C) \mid u_{\restriction A,B} \in \sigma \wedge u_{\restriction B,C} \in \tau\}$, composition is then defined as $\sigma; \tau = \{u_{\restriction A,C} \mid u \in \sigma || \tau\}$. Composition is associative and admits copycat strategies as identities.

***P*-visible strategies.** A strategy σ is ***P*-visible** if each of its moves points to the current P-view. Formally, for all $sab \in \sigma$, b points inside $\ulcorner sa \urcorner$. P-visible strategies are stable under composition, as is proved for instance in [13]. They correspond loosely to functional programs with ground type references [1].

Innocent strategies. The class of *innocent* strategies is central in game semantics, because of their correspondence with purely functional programs (or λ-terms) and of their useful definability properties. A strategy σ is **innocent** if

$$sab \in \sigma \wedge t \in \sigma \wedge ta \in \mathcal{L}_A \wedge \ulcorner sa \urcorner = \ulcorner ta \urcorner \Rightarrow tab \in \sigma$$

Intuitively, an innocent strategy only takes its P-view into account to determine its next move. Indeed, any innocent strategy is characterized by a set of P-views. This observation is very important since P-views can be seen as abstract representations of branches of η-expanded Böhm trees (*a.k.a.* Nakajima trees [23]) : this is the key to the definability process on innocent strategies [16]. It is

[3] In the terminology of [13], it is the *long* O-view.

quite technical to prove that innocent strategies are stable under composition, proofs can be found for instance in [13, 5]. Arenas and innocent strategies form a cartesian closed category and are therefore a model of simply typed λ-calculus.

Bounded strategies. A strategy σ is **bounded** if it is P-visible and if the length of its P-views is bounded: formally, there exists $N \in \mathbb{N}$ such that for all $s \in \sigma$, $|\ulcorner s \urcorner| \leq N$. Bounded strategies are stable under composition, as is proved in [6] for the innocent case and in [5] for the general case. This result corresponds loosely to the strong normalisation result on simply-typed λ-calculus. Syntactically, bounded strategies include the interpretation of all terms of a functional programming language without a fixed point operator but with Algol-like ground type references (for details about how reference cells get interpreted as strategies see for instance [1], it is obvious that this interpretation yields a bounded strategy) and arbitrary non determinism. This remark is important since it implies that our results will hold for any program written with these constructs, as long as they do not use recursion or a fixed point operator.

2.3 Size of strategies and interactions

Since in this paper we will be interested in the length of interactions, it is sensible to make it precise first what we mean by the *size* of strategies. Let σ be a bounded strategy, its size is defined as

$$|\sigma| = \frac{max_{s \in \sigma} |\ulcorner s \urcorner|}{2}$$

All our analysis on the size of interactions will be based on this notion of size of strategies. Our starting point is the following finiteness result, proved in [6]. We say that an interaction $u \in I(A, B, C)$ is **passive** if the only move by the external Opponent on A, C is the initial move on C, so that the interaction stops as soon as we need additional input from the external Opponent.

Proposition 1. *Let $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ be bounded strategies and let $u \in \sigma || \tau$ be a passive interaction, then u is finite.*

Using this, we can actually deduce the existence of an uniform bound on the length of such $u \in \sigma || \tau$, which only depends on the respective size of σ and τ:

Lemma 1. *For all $n, p \in \mathbb{N}$ there is a lesser $N(n, p) \in \mathbb{N}$ such that for all arenas A, B and C, for all $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ such that $|\sigma| \leq p$ and $|\tau| \leq n$, for all passive $u \in \sigma || \tau$ we have $|u| \leq N(n, p)$.*

Proof. For arenas A, B and C consider the set $T_{A,B,C}$ of all passive interactions $u \in I(A, B, C)$ such that for all $s \sqsubseteq u_{\restriction B,C}$, $|\ulcorner s \urcorner| \leq 2n$ and for all $s \sqsubseteq u_{A,B}$, $|\ulcorner s \urcorner| \leq 2p$. Then, consider the union T of all the $T_{A,B,C}$, our goal here is to find a bound on the length of all elements of T. Consider now the tree structure on T given by the prefix ordering. To make this tree finitely branching, consider the relation $m \cong n \Leftrightarrow depth(m) = depth(n)$ on moves, where $depth(m)$ is the number of pointers required to go from m to an initial move. The tree $T/\cong$ is now finitely branching, but is also well-founded by Proposition 1, therefore it is finite by König's lemma[4]. Let $N(n,p)$ be its maximal depth, it is now obvious that it satisfies the required properties. $\square$

We have proved the existence of the uniform bound $N(n,p)$, but in a way that provides no feasible means of estimating $N(n,p)$. The goal of the rest of this paper is to estimate this bound as precisely as possible. As a matter of fact, we will be mainly interested in the "typed" variant $N_d(n,p)$, defined as the maximum length of all possible passive interactions between strategies $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ of respective size p and n, where B has a finite depth $d-1$.

3 Pointer structures and rewriting

We have seen that to prove Lemma 1, we must consider plays up to an equivalence relation $\cong$ which assimilates all moves at the same depth. Indeed, general arenas and plays contain information which is useless for us. Following [6], we will here reason on *pointer structures*, which result of considering moves in plays up to $\cong$. Pointer structures are also similar to the *parity pointer functions* of Harmer, Hyland and Melliès [14] and to the *interaction sequences* of Coquand [7]. We will delve here into their combinatorics and extract from them a small rewriting system, whose study is sufficient to characterize their length.

3.1 $N_d(n,p)$ as a bound for pointer structures

Visible pointer structures. In [6], we introduced pointer structures by elementary axioms, independent of the general notions of game semantics. Instead here, we define **pointer structures** as usual alternating plays, but on the particular "pure" arena $I_\omega = \bigsqcup_{n\in\mathbb{N}} I_n$, where $I_0 = \perp$ ($\perp$ is the singleton arena with just one Opponent move) and $I_{n+1} = I_n \Rightarrow \perp$. As we are interested in the interaction between P-visible strategies, we will only consider **visible** pointer structures, where both players point in their corresponding view. Formally, s is visible if for all $s'p \sqsubseteq^P s$, a points inside $\ulcorner s' \urcorner$ and if for all $s'o \sqsubseteq^O s$, o points inside $\llcorner s' \lrcorner$. The **depth** of a visible pointer structure s is the smallest d such that s is a play on I_d. Let us denote by $\mathcal{V}$ the set of all visible pointer structures.

Atomic agents. After forgetting information on plays, let us forget information on strategies. Instead of considering bounded strategies with all their intentional behaviour, we will just keep the data of their size. Pointer structures will then be considered as interactions between the corresponding numbers

[4]Or, more adequately, the fan theorem.

which will be called **atomic agents**. If n is such a natural number, we define its **trace** as follows, along with the dual notion of **co-trace**:

$$\begin{array}{rcl} Tr(n) & = & \{s \in \mathcal{V} \mid \forall s' \sqsubseteq s, |\ulcorner s' \urcorner| \leq 2n\} \\ coTr(p) & = & \{s \in \mathcal{V} \mid \forall s' \sqsubseteq s, |\llcorner s' \lrcorner| \leq 2p+1\} \end{array}$$

An **interaction** at depth d between n and p is a visible pointer structure s of depth at most d such that $s \in Tr(n) \cap coTr(p)$. We write $s \in n \star_d p$. These definitions allow to give the following strategy-free equivalent formulation of $N_d(n, p)$.

Lemma 2. *Let n and p be natural numbers and $d \geq 2$, then*

$$N_d(n, p) = max\{|s| \mid s \in n \star_d p\}$$

Proof. Consider the maximal bounded strategies of respective size n and p, defined as $\mathbf{n} = \{s \in I_d \mid \forall s' \sqsubseteq s, |\ulcorner s' \urcorner| \leq 2n$ and $\mathbf{p} = \{s \in I_{d-1} \mid \forall s' \sqsubseteq s, |\ulcorner s' \urcorner| \leq 2p$. Then pointer structures in $n \star_d p$ are the same as (passive) interactions in $\mathbf{p}||\mathbf{n}$, thus $max\{|s| \mid s \in n \star_d p\} \leq N_d(n, p)$. Reciprocally, if $\sigma : A \Rightarrow B$ has size p and $\tau : B \Rightarrow C$ has size n and if $u \in \sigma||\tau$ is passive, then if u' denotes u where moves are considered up to $\cong$ we have $u' \in \mathbf{p}||\mathbf{n}$ thus $u' \in n \star_d p$ and $N_d(n, p) = max\{|s| \mid s \in n \star_d p\}$. $\square$

3.2 Agents

To bound the length of a pointer structure s, our idea is to label each of its moves s_i by an object t, expressing the size that the strategies have left. Let us consider here an analogy between pointer structures and the execution of λ-terms by the KAM[5]. Consider the following three KAM computation steps:

$$(\lambda x.xS) \star T \cdot \pi_0 \quad \rightsquigarrow^3 \quad T \star S^{x \mapsto T} \cdot \pi_0$$

The interaction between two closed terms (with empty environment) leads, after three steps of computation, to the interaction between two *open terms* T and S (where x is free in S), with an environment. By analogy, if s_0 is labelled by the pair (n, p) of interacting "strategies", each move s_i should correspond to an interaction between objects (a, b), where a and b have a tree-like structure which is reminiscent of those of closures[6].

We will call a **pointed visible pointer structure** (pvps) a pair (s, i) where s is a visible pointer structure and $i \leq |s| - 1$ is an arbitrary "starting" move. We adapt the notions of size and depth for pvps, and introduce a notion of *context*.

Definition 1. *Let (s, i) be a pointed visible pointer structure. The* ***residual size*** *of s at i, written* rsize(s, i)*, is defined as follows:*

[5]The syntax used here seems natural enough, but is for instance described in [5].

[6]As in the example above, closures are pairs M^σ where M is an open term and σ is an *environment*, *i.e.* a mapping which to each free variable of M associates a closure.

- *If s_i is an Opponent move, it is* $\max_{s_i \in \ulcorner s_{\leq j} \urcorner} |\ulcorner s_{\leq j} \urcorner| - |\ulcorner s_{\leq i} \urcorner| + 1$
- *If s_i is a Player move, it is* $\max_{s_i \in \llcorner s_{\leq j} \lrcorner} |\llcorner s_{\leq j} \lrcorner| - |\llcorner s_{\leq i} \lrcorner| + 1$

where $s_i \in \ulcorner s_{\leq j} \urcorner$ means that the computation of $\ulcorner s_{\leq j} \urcorner$ reaches[7] *s_i. Dually, we have the notion of **residual co-size** of s at i, written* $\mathrm{rcosize}(s, i)$*, defined as follows:*

- *If s_i is an Opponent move, it is* $\max_{s_i \in \llcorner s_{\leq j} \lrcorner} |\llcorner s_{\leq j} \lrcorner| - |\llcorner s_{\leq i} \lrcorner| + 1$
- *Otherwise,* $\max_{s_i \in \ulcorner s_{\leq j} \urcorner} |\ulcorner s_{\leq j} \urcorner| - |\ulcorner s_{\leq i} \urcorner| + 1$

The residual depth *of s at i is the maximal length of a pointer chain in s starting from s_i.*

Definition 2. *Let s be a visible pointer structure. We define the **context** of (s, i) as:*

- *If s_i is an O-move, the set $\{s_{n_1}, \dots, s_{n_p}\}$ of O-moves appearing in $\ulcorner s_{<i} \urcorner$,*
- *If s_i is a P-move, the set $\{s_{n_1}, \dots, s_{n_p}\}$ of P-moves appearing in $\llcorner s_{<i} \lrcorner$.*

*In other words it is the set of moves to which s_{i+1} can point whilst abiding to the visibility condition, except s_i. We also need the dual notion of co-context, which contains the moves the other player can point to. The **co-context** of (s, i) is:*

- *If s_i is an O-move, the set $\{s_{n_1}, \dots, s_{n_p}\}$ of P-moves appearing in $\llcorner s_{<i} \lrcorner$,*
- *If s_i is a P-move, the set $\{s_{n_1}, \dots, s_{n_p}\}$ of O-moves appearing in $\ulcorner s_{<i} \urcorner$.*

Definition 3. *A **general agent** (just called agent for short) is a finite tree, whose nodes and edges are both labelled by natural numbers. If $a_1, \dots, a_p$ are agents and $d_1, \dots, d_p$ are natural numbers, we write:*

$$n[\{d_1\}a_1, \dots, \{d_p\}a_p] = \begin{array}{ccc} & n & \\ {}^{d_1}\swarrow & & \searrow^{d_p} \\ a_1 & \cdots & a_p \end{array}$$

Definition 4 (Trace, co-trace, interaction)**.** *Let us generalize the notion of trace to general agents. The two notions Tr and $coTr$ are defined by mutual recursion, as follows: let $a = n[\{d_1\}a_1, \dots, \{d_p\}a_p]$ be an agent. We say that (s, i) is a **trace** (resp. a **co-trace**) of a, denoted $(s, i) \in Tr(a)$ (resp. $(s, i) \in coTr(a)$) if the following conditions are satisfied:*

- $\mathrm{rsize}(s, i) \leq 2n$ *(resp.* $\mathrm{rcosize}(s, i) \leq 2n + 1$*),*
- *If $\{s_{n_1}, \dots, s_{n_p}\}$ is the context of (s, i) (resp. co-context), then for each $k \in \{1, \dots, p\}$ we have $(s, n_k) \in coTr(a_k)$.*

[7] So starting from s_j and following Opponent's pointers eventually reaches s_i.

- *If $\{s_{n_1}, \ldots, s_{n_p}\}$ is the context of (s,i) (resp. co-context), then for each $k \in \{1, \ldots, p\}$ the residual depth of s at n_k is less than d_k.*

Then, we define an ***interaction*** *of two agents a and b at depth d as a pair $(s,i) \in Tr(a) \cap coTr(b)$ where the residual depth of s at i is less than d, which we write $(s,i) \in a \star_d b$.*

Notice that we use the same notations Tr, $coTr$ and $\star$ both for natural numbers and general agents. This should not generate any confusion, since the definitions just above coincide with the previous ones in the special case of "atomic", or closed, agents: if n and p are natural numbers, then obviously $s \in n \star_d p$ if and only if $(s,0) \in n[] \star_d p[]$. Note also that definitions are adapted here to this particular setting where strategies are replaced by natural numbers, however they could be generalized to the usual notion of strategies. An agent would be then a tree of strategies, and a trace of this agent would be a possible interaction between all these strategies. This would be a new approach to the problem of *revealed* or *uncovered* game semantics [12, 4], where strategies are not necessarily cut-free.

3.3 Simulation of visible pointer structures

We introduce now the main tool of this paper, a reduction on agents which "simulates" visible pointer structures: if $n[\{d_1\}a_1, \ldots, \{d_p\}a_p]$ and b are agents ($n > 0$), we define the non-deterministic reduction relation $\rightsquigarrow$ on triples (a, d, b), where d is a depth (a natural number) and a and b are agents, by the following two cases:

$$\begin{array}{lcl} (n[\{d_1\}a_1, \ldots, \{d_p\}a_p], d, b) & \rightsquigarrow & (a_i, d_i - 1, (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p, \{d\}b]) \\ (n[\{d_1\}a_1, \ldots, \{d_p\}a_p], d, b) & \rightsquigarrow & (b, d-1, (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p, \{d\}b]) \end{array}$$

where $i \in \{1, \ldots, p\}$, $d_i > 0$ in the first case and $d > 0$ in the second case. We can now state the following central proposition.

Proposition 2 (Simulation)**.** *Let $(s,i) \in a \star_d b$, then if s_{i+1} is defined, there exists $(a, d, b) \rightsquigarrow (a', d', b')$ such that $(s, i+1) \in a' \star_{d'} b'$.*

Proof. The proof proceeds by a close analysis of where in its P-view (resp. O-view) s_{i+1} can point. If it points to s_i, then the active strategy asks for its argument which corresponds to the second reduction case. If it points to some element s_{n_i} of its context, the active strategy calls the i-th element of its context: this is the first reduction case, putting the subtree a_i in head position. The rest of the proof consists in technical verifications, to check that the new triple (a', d', b') is such that $(s, i+1) \in a' \star_{d'} b'$. □

The result above will be sufficient for our purpose. Let us mention in passing that the connection between visible pointer structures and agents is in fact tighter: a reduction chain starting from a triple $(n[], d, p[])$ can also be canonically mapped to a pointed visible pointer structure in $n \star_d p$, and the two translations are inverse of one another. The interested reader is directed to [5].

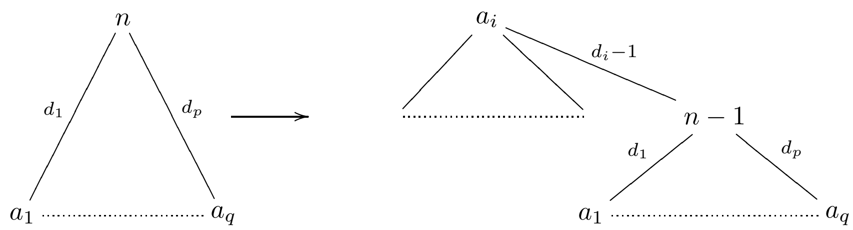

Figure 1: Rewriting rule on agents

Before going on to the study of the rewriting rules introduced above, let us give a last simplification. If $a = n[\{d_1\}t_1, \ldots, \{d_p\}t_q]$ and b are agents, then $a \cdot_d b$ will denote the agent obtained by appending b as a new son of the root of a with label d, *i.e.* $n[\{d_1\}t_1, \ldots, \{d_p\}t_q, \{d\}b]$. Consider the following non-deterministic rewriting rule on agents:

$$n[\{d_1\}a_1, \ldots, \{d_p\}a_p] \rightsquigarrow a_i \cdot_{d_i-1} (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p]$$

Both rewriting rules on triples (a, d, b) are actually instances of this reduction, by the isomorphism $(a, d, b) \mapsto a \cdot_d b$. We let the obvious verification to the reader. This is helpful, as all that remains to study is this reduction on agents illustrated in Figure 1. To summarize, if $N(a)$ denotes the length of the longest reduction sequence starting from an agent a, we have the following property.

Proposition 3. *Let $n, p \geq 0$, $d \geq 2$, then $N_d(n, p) \leq N(n[\{d\}p[]]) + 1$.*

Proof. Obvious from the simulation lemma, adding 1 for the initial move which is not accounted for by the reduction on agents. In fact this is an equality, as one can prove using the *reverse* simulation lemma mentioned above. See [5]. □

4 Length of interactions

The goal of this section is to study the reduction on agents introduced above, and to estimate its maximal length. We will first provide an upper bound for this length, adapting a method used by Beckmann [3] to estimate the maximal length of reductions on simply typed λ-calculus. We will then discuss the question of lower bounds, and finally describe an application to head linear reduction.

4.1 Upper bound

We define on agents a predicate $\vert\frac{\alpha}{\rho}$, which introduction rules are compatible both with syntax and reduction.

Definition 5. *The predicate* $\vdash^{\alpha}_{\rho}$ *(where* ρ, α *range over natural numbers) is defined on agents in the following inductive way.*

- BASE. $\vdash^{\alpha}_{\rho} 0[\{d_1\}a_1, \ldots, \{d_p\}a_p]$
- RED. *Suppose* $a = n[\{d_1\}a_1, \ldots, \{d_p\}a_p]$. *Then if for all* a' *such that* $a \rightsquigarrow a'$ *we have* $\vdash^{\alpha}_{\rho} a'$ *and if we also have* $\vdash^{\alpha}_{\rho} (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p]$, *then* $\vdash^{\alpha+1}_{\rho} a$.
- CUT. *If* $\vdash^{\alpha}_{\rho} a$, $\vdash^{\beta}_{\rho} b$ *and* $d \leq \rho$, *then* $\vdash^{\alpha+\beta}_{\rho} a \cdot_d b$.

By this inductive definition, each proposition $\vdash^{\alpha}_{\rho} a$ is witnessed by a tree using BASE, RED and CUT. RED-free trees look like syntax trees, are easy to build but give few information on the reduction, whereas CUT-free trees look like reduction trees, are difficult to build but give very accurate information on the length of reduction. The idea of the proof is then to design an automatic way to turn a RED-free tree to a CUT-free tree, *via* a cut elimination lemma. Let us now give the statement and sketch the proof of the four important lemmas that underlie our reasoning.

A **context-agent** $a()$ is a finite tree whose edges are labelled by natural numbers, and whose nodes are labelled either by natural numbers, or by the variable x, with the constraint that all edges leading to x must be labelled by the same number d; d is called the **type** of x in $a()$. If We denote by $a(b)$ the result of substituting of all occurrences of x in $a()$ by b. We denote by $a(\emptyset)$ the agent obtained by deleting in a all occurrences of x, along with the edges leading to them.

Lemma 3 (Substitution lemma)**.** *If* $\vdash^{\alpha}_{\rho} a(\emptyset)$, $\vdash^{\beta}_{\rho} b$ *and* $d \leq \rho + 1$ *(where* d *is the type of* x *in* a*), then* $\vdash^{\alpha(\beta+1)}_{\rho} a(b)$

Proof. We prove by induction on the tree witness for $\vdash^{\alpha}_{\rho} a(\emptyset)$ that the above property is true for all context-arena $a'()$ such that $a(\emptyset) = a'(\emptyset)$. The way to handle each case is essentially forced by the induction hypothesis. □

Lemma 4 (Cut elimination lemma)**.** *Suppose* $\vdash^{\alpha}_{\rho+1} a$. *Then if* $\alpha = 0$, $\vdash^{0}_{\rho} a$. *Otherwise,* $\vdash^{2^{\alpha-1}}_{\rho} a$.

Proof. By induction on the witness for $\vdash^{\alpha}_{\rho+1} a$, using the substitution lemma when the last rule is CUT with a type of $\rho + 1$. □

Lemma 5 (Recomposition lemma)**.** *Let* a *be an agent. Then* $\vdash^{max(a)|a|}_{depth(a)} a$, *where* $depth(a)$ *is the maximal label of an edge in* a, $max(a)$ *is the maximal label of a node and* $|a|$ *is the number of nodes.*

Proof. By induction on a. □

Lemma 6 (Bound lemma)**.** *Let* a *be an agent, then if* $\vdash^{\alpha}_{0} a$, $N(a) \leq \alpha$.

Proof. The only used rules are Base, Red and Cut with $\rho = 0$. These Cut rules do not add any possible reduction and are easy to eliminate, then the lemma is easily proved by induction on a. $\square$

These lemmas are sufficient to give a first upper bound, by iterating the cut elimination lemma starting from the witness tree for $\vdash^{\alpha}_{\rho} a$ generated by the recomposition lemma. However when the type is small, some of the lemmas above can be improved. For instance if $\vdash^{\alpha}_{0} a(\emptyset)$, $\vdash^{\beta}_{0} b$ and the type of x in $a()$ is 1, then $\vdash^{\alpha+\beta}_{0} a(b)$, since once the reduction reaches b it will never enter $a()$ again. Using this we get a "base" cut-elimination lemma, stating that for all a, whenever $\vdash^{\alpha}_{1} a$ then we have actually $\vdash^{\alpha}_{0} a$ instead of $\vdash^{2^{\alpha-1}}_{0} a$. Using this, we prove the following.

Theorem 1 (Upper bound)**.** *Let $depth(a)$ denote the highest edge label in a, $max(a)$ means the highest node label and $|a|$ means the number of nodes of a. Then if $depth(a) \geq 1$ and $max(a) \geq 1$ we have:*

$$N(a) \leq 2^{max(a)|a|-1}_{depth(a)-1}$$

For the particular case when $a = n[\{d\}p[]]$ and if $d \geq 2$ we have:

$$N_d(n,p) \leq 2^{n(p+1)}_{d-2}$$

Proof. Both proofs are rather direct. For the first part, by the recomposition lemma we have $\vdash^{max(a)|a|}_{depth(a)} a$. It suffices then to apply $depth(a) - 1$ times the cut elimination lemma, then use the "base" cut-elimination lemma to eliminate the remaining cuts. For the second part we reason likewise, but rely on the substitution lemma instead of the recomposition lemma to get $\vdash^{n(p+1)}_{d-1} n[\{d\}p[]]$, which gives $N(n[\{d\}p[]]) \leq 2^{n(p+1)-1}_{d-2}$. But we have $N_d(n,p) \leq N(n[\{d\}p[]]) + 1$ by Proposition 3, which concludes the proof. $\square$

Note that whereas the bounds in [3] are asymptotic and give poor quantitative information if instantiated on small types, our bound does provide valuable information on interactions with small depth. For instance, if $\sigma : A \Rightarrow B$ and $\tau : B \Rightarrow C$ such that $\mathrm{rsize}(\sigma) = p$, $\mathrm{rsize}(\tau) = n$ and the depth of B is at most 2, then no interaction between σ and τ can be longer than $N_3(n,p) \leq 2^{n(p+1)}$. As we will see below, this can not be significantly improved. In fact, we conjecture that for all $n \geq 1$ and $p \geq 2$, we have $N_3(n,p) = 2\frac{p^n-1}{p-1} + 1$: this was found and machine-checked for all $n + p \leq 17$ thanks to an implementation of agents and their reduction, unfortunately we could not prove its correctness, nor generalize it to higher depths.

4.2 Lower bound

As argued in the introduction, the upper bound above applies to several programming languages executed by head linear reduction, possibly featuring non determinism and/or ground type references, therefore the fact that we used game semantics to prove it increases its generality. On the other hand, if we try to give the closest possible lower bound for $N_d(n,p)$ using the full power of visible pointer structures, we would get a lower bound without meaning for most languages concerned by the upper bound, since pointer structures have no innocence or determinism requirements[8]. Therefore what makes more sense is to describe a lower bound in the more restricted possible framework, *i.e.* simply typed λ-calculus.

We won't detail the construction much, as the method is standard and does not bring a lot to our analysis. The idea is to define higher types for church integers by $A_0 = \perp$ and $A_{n+1} = A_n \to A_n$. Then, denoting by $\underline{n}_p$ the church integer for n of type A_{p+2}, we define $S_n = \underline{2}_n\underline{2}_{n-1}\dots\underline{2}_0 : A_2$. We apply then S_n to $id_\perp$ to get a term whose head linear reduction chain has at least 2^1_{n+1} steps. In game semantics, $[\![\underline{2}_n]\!]$ has size $n+3$ and all other components have size smaller than $n+2$, the depth of the ambient arena being $n+2$. The function $N_d(n,p)$ being monotonically increasing in all its parameters we have the following inequalities for $3 \leq d \leq min(n-1,p)$, both bounds making sense for all programming languages containing the simply-typed λ-calculus and whose terms can be interpreted as bounded strategies.

$$2^2_{d-2} \leq N_d(n,p) \leq 2^{n(p+1)}_{d-2}$$

Note that from this we can deduce bounds for $N(n,p)$, when we have no information on the depth of the ambient arena. Indeed, we always have $d \leq 2n$ and $d \leq 2p+1$ because a pointer chain in a play is visible by both players. Thus, $N(n,p) = N_{min(2n,2p+1)}(n,p)$.

4.3 Application to head linear reduction

Earlier works on game semantics [8] suggest that in every games model of a programming language lies a hidden notion of linear reduction, head linear reduction when modelling call-by-name evaluation: this is the foundation for our claim that our game-theoretic result is really about the length of execution in programming languages whose terms can be described as bounded strategies. Of course it requires some work to interface execution in these programming languages to our game-theoretic results, and part of this work has to be redone in each case. To illustrate this, we now describe how to extract from our results a theorem about the length of head linear reduction sequences in simply-typed λ-calculus. For the formal definition of head linear reduction, the reader is directed to [9]. If S is a λ-term then the **spinal height** of S is the quantity $sh(S)$ defined by induction as $sh(x) = 1$, $sh(\lambda x.S) = sh(S)$ and $sh(ST) = max(sh(S), sh(T)+1)$; when S is a $\beta\eta$-normal form, $sh(S)$ is nothing but the height of its Böhm tree. The **height** of S is the subtly dif-

[8]Our experiments with pointer structures and agents confirmed indeed that the possibility to use non-innocent behaviour does allow significantly longer plays.

ferent quantity[9] $h(S)$ defined by $h(x) = 1$, $h(\lambda x.M) = h(M)$ and $h(MN) = max(h(M), h(N)) + 1$. Finally, the **level** of a type $lv(A)$ is defined by $lv(\bot) = 0$ and $lv(A \to B) = max(lv(A) + 1, lv(B))$ and the **degree** $g(S)$ of a term is the maximal level of the type of all subterms of S.

A **game situation** [5] is the data of λ-terms $S : A_1 \to \ldots \to A_p \to B$ and $T_1 : A_1, \ldots T_p : A_p$ in η-long β-normal form, and we are interested in the term $ST_1 \ldots T_p$. Our game-theoretic results apply immediately to game situations, because of the connection between game-theoretic interaction and head linear reduction [8]: if $N(ST_1 \ldots T_p)$ denotes the length of the head linear reduction chain of $ST_1 \ldots T_p$, then we have $N(ST_1 \ldots T_p) \leq N_d(n, p)$ where d is the depth of the arena corresponding to $A \to B$, n is the size of $[\![S]\!]$ and p is the maximal size of all of the $[\![T_i]\!]$. But since S and T_i are already in η-long β-normal form, we have $|[\![S]\!]| = sh(S)$ and $|[\![T_i]\!]| = sh(T_i)$. Thus, we conclude that in the case of a game situation we have:

$$N(ST_1 \ldots T_p) \leq 2^{sh(S)(max_i(sh(T_i))+1)}_{max_i lv(A_i)-1}$$

Outside of game situations, it is less obvious to see how our results apply. The more elegant approach would be probably to extend the connection between head linear reduction and game semantics to *revealed* game semantics, which would give the adequate theoretical foundations to associate an agent to any η-long λ-term. Without these tools, we can nonetheless apply the following hack. Suppose we have a λ-term S. The idea is to "delay" all redexes, replacing each redex $(\lambda x.S)T$ of type $A \to B$ in S with $y_{A,B}(\lambda x.S)T$, where we add a new symbol $y_{A,B} : (A \to B) \to A \to B$ for each pair (A, B). We iterate this operation until we reach a β-normal λ-term S^t, which satisfies $sh(S^t) \leq h(S)$. We then expand S^t to its η-long form $\eta(S^t)$, which satisfies $sh(\eta(S^t)) \leq sh(S^t) + g(S^t) \leq h(S) + g(S) + 1$. We consider now the term $(\lambda y_1 \ldots y_p.\eta(S^t))ev_1 \ldots ev_p$, where each y_i binds one of the new symbols $y_{A,B}$, and $ev_i : (A \to B) \to A \to B$ is the (η-long form of) the corresponding evaluation λ-term. We recognise here a game situation, whose head linear reduction chain is necessarily longer than for S (we have only added steps due to the delaying of redexes and η-expansion). Using the inequality above for game situations, we conclude:

$$N(S) \leq 2^{(h(S)+g(S)+1)(g(S)+1)}_{g(S)}$$

5 Conclusion & future work

Applied to head linear reduction on simply typed λ-calculus, our results show that the price of linearity is not as high as one might expect. Not only the bounds remain in $\mathcal{E}^4$, but they are only slightly higher than those for usual β-reduction: in particular, the height of the tower of exponentials is the same.

[9] One can easily prove that on closed terms, it is always less than the more common notion of height defined as $h(x) = 0$, $h(\lambda x.S) = 1 + h(S)$ and $h(ST) = max(h(S), h(T)) + 1$, for which our upper bound consequently also holds.

A strength of our method is that it is not restricted to λ-calculus; the results should indeed immediately apply as well to similar notions of reduction on other total programming languages. Beyond ground type references and non determinism, there are also games model of call-by-value languages [2] generating pointer structures as well, thus this work should also provide bounds for the corresponding call-by-value linear reduction (*tail* linear reduction?). All the tools used here also can be extended to *non-alternating plays* [20], which suggests that this work could be used to give bounds to the length of reductions in some restricted concurrent languages.

We also believe *agents* are worth studying further. Their combinatorial nature and their connection to execution of programs may prove interesting for the study of higher order systems with restricted complexity, such as *light* linear logics [11]. For instance, proofs typable in light systems may correspond to agents with some restricted behaviours, which would make them a valuable tool for the study of programming languages with implicit complexity.

Acknowledgements. This work was partially supported by the French ANR project CHOCO. The author also would like to thank Fabien Renaud for interesting discussions on related subjects.

References

[1] Samson Abramsky and Guy McCusker. Linearity, Sharing and State: a Fully Abstract Game Semantics for Idealized Algol with active expressions, 1997.

[2] Samson Abramsky and Guy McCusker. Call-by-value games. In Mogens Nielsen and Wolfgang Thomas, editors, *6th Annual Conference of the European Association for Computer Science Logic*, volume 1414 of *Lecture Notes in Computer Science*. Springer, 1998.

[3] A. Beckmann. Exact bounds for lengths of reductions in typed λ-calculus. *Journal of Symbolic Logic*, 66(3):1277–1285, 2001.

[4] W. Blum. *Thesis fascicle: Local computation of β-reduction.* PhD thesis, University of Oxford, 2008.

[5] Pierre Clairambault. *Logique et Interaction : une Étude Sémantique de la Totalité.* PhD thesis, Université Paris Diderot, 2010.

[6] Pierre Clairambault and Russ Harmer. Totality in arena games. *Annals of Pure and Applied Logic*, 2009.

[7] Thierry Coquand. A semantics of evidence for classical arithmetic. *Journal of Symbolic Logic*, 60(1):325–337, 1995.

[8] Vincent Danos, Hugo Herbelin, and Laurent Regnier. Game semantics and abstract machines. In *11th IEEE Symposium on Logic in Computer Science*, pages 394–405, 1996.

[9] Vincent Danos and Laurent Regnier. How abstract machines implement head linear reduction. Unpublished, 2003.

[10] N.G. de Bruijn. Generalizing Automath by means of a lambda-typed lambda calculus. *Mathematical Logic and Theoretical Computer Science*, 106:71–92, 1987.

[11] Jean-Yves Girard. Light linear logic. *Inf. Comput.*, 143(2):175–204, 1998.

[12] W. Greenland. *Game semantics for region analysis.* PhD thesis, University of Oxford, 2004.

[13] Russ Harmer. Innocent game semantics. *Lecture notes*, 2004–2007.

[14] Russ Harmer, Martin Hyland, and Paul-André Melliès. Categorical combinatorics for innocent strategies. In *IEEE Symposium on Logic in Computer Science*, pages 379–388, 2007.

[15] Russ Harmer and Guy McCusker. A fully abstract game semantics for finite nondeterminism. In *IEEE Symposium on Logic in Computer Science*, pages 422–430, 1999.

[16] Martin Hyland and C.-H. Luke Ong. On full abstraction for PCF: I, II and III. *Information and Computation*, 163(2):285–408, December 2000.

[17] Jean-Louis Krivine. Un interpréteur du λ-calcul. *Unpublished*, 1985.

[18] Ugo Dal Lago and Olivier Laurent. Quantitative game semantics for linear logic. In *CSL*, pages 230–245, 2008.

[19] James Laird. Full abstraction for functional languages with control. In *IEEE Symposium on Logic in Computer Science*, pages 58–67, 1997.

[20] James Laird. A game semantics of the asynchronous π-calculus. In *CONCUR*, pages 51–65, 2005.

[21] Gianfranco Mascari and Marco Pedicini. Head linear reduction and pure proof net extraction. *Theoretical Computer Science*, 135(1):111–137, 1994.

[22] Guy McCusker. Games and Full Abstraction for FPC. *Information and Computation*, 160(1-2):1–61, 2000.

[23] R. Nakajima. Infinite normal forms for the λ-calculus. *λ-Calculus and Computer Science Theory*, pages 62–82, 1975.

[24] H. Schwichtenberg. Complexity of normalization in the pure typed lambda-calculus. *Studies in Logic and the Foundations of Mathematics*, 110:453–457, 1982.

A pointer structures and rewriting

Lemma 7. *Let s be a pointed visible pointer structure and $a = n[\{d_1\}a_1, \dots, \{d_p\}a_p]$ an agent such that $(s,i) \in coTr(a)$. Then if $s_j \to s_i$, $(s,j) \in Tr(a)$.*

Proof. Let us suppose without loss of generality that s_i is an Opponent move; the other case can be obtained just by switching Player/Opponent and P-views/O-views everywhere. Then s_j being a Player move, we have to check first that $\mathrm{rsize}(s,j) \leq 2n$, *i.e.*

$$\max_{s_j \in \llcorner s_{\leq k} \lrcorner} |\llcorner s_{\leq k} \lrcorner| - |\llcorner s_{\leq j} \lrcorner| + 1 \leq 2n$$

We use that $\mathrm{rcosize}(s,i) \leq 2n+1$, *i.e.*

$$\max_{s_i \in \llcorner s_{\leq k} \lrcorner} |\llcorner s_{\leq k} \lrcorner| - |\llcorner s_{\leq i} \lrcorner| + 1 \leq 2n+1$$

But $s_j \to s_i$, hence $|\llcorner s_{\leq j} \lrcorner| = |\llcorner s_{\leq i} \lrcorner| + 1$ and the inequality is obvious. We need now to examine the context of (s,j). Since s_j is a Player move, it is defined as the set $\{s_{n_1}, \dots, s_{n_p}\}$ of Player moves appearing in $\llcorner s_{<j} \lrcorner$, which is also the set of Player moves appearing in $\llcorner s_{<i} \lrcorner$ and therefore the co-context of (s,i). But $(s,i) \in coTr(a)$, hence for all $k \in \{1, \dots, p\}$ we have $(s, n_k) \in coTr(a_k)$ which is exactly what we needed. □

Proposition 4 (Simulation)**.** *Let $(s,i) \in a \star_d b$, then if s_{i+1} is defined, there exists $(a,d,b) \rightsquigarrow (a',d',b')$ such that $(s,i+1) \in a' \star_{d'} b'$.*

Proof. Suppose $a = n[\{d_1\}a_1, \dots, \{d_p\}a_p]$. Let $\{s_{n_1}, \dots, s_{n_p}\}$ be the context of (s,i). By visibility, s_{i+1} must either point to s_i or to an element of the context. Let us distinguish cases.

- If $s_{i+1} \to s_i$, then we claim that $(s,i+1) \in b \star_{d-1} (n-1)[\{d_1\}a_1, \dots, \{d_p\}a_p, \{d\}b]$, *i.e* $(s,i+1) \in Tr(b)$, $(s,i+1) \in coTr((n-1)[\{d_1\}a_1, \dots, \{d_p\}a_p, \{d\}b])$ and the depth of s relative to $i+1$ is at most $d-1$. For the first part, we use that $(s,i) \in a \star_d b$: in particular, $(s,i) \in coTr(b)$ and since $s_{i+1} \to s_i$ this implies by Lemma 7 that $(s,i+1) \in Tr(b)$. For the second part, we must first check that $\mathrm{rcosize}(s,i+1) \leq 2(n-1)+1$. Let us suppose without loss of generality that s_i is an Opponent move, all the reasoning below can be adapted by switching Player/Opponent and P-views/O-views everywhere. We want to prove:

  $$\mathrm{rcosize}(s,i+1) = \max_{s_{i+1} \in \ulcorner s_{\leq j} \urcorner} |\ulcorner s_{\leq j} \urcorner| - |\ulcorner s_{\leq i+1} \urcorner| + 1 \leq 2(n-1)+1$$

  But since $(s,i) \in Tr(a)$, we already know:

  $$\mathrm{rsize}(s,i) = \max_{s_i \in \ulcorner s_{\leq j} \urcorner} |\ulcorner s_{\leq j} \urcorner| - |\ulcorner s_{\leq i} \urcorner| + 1 \leq 2n$$

  Thus we only need to remark that $|\ulcorner s_{\leq i+1} \urcorner| = |\ulcorner s_{\leq i} \urcorner| + 1$ since s_{i+1} is a Player move. Now, we must examine the co-context of $(s,i+1)$, but by definition of P-view it is $\{s_{n_1}, \ldots, s_{n_p}, s_i\}$ where $\{s_{n_1}, \ldots, s_{n_p}\}$ is the context of (s,i). Since $(s,i) \in Tr(n[a_1, \ldots, a_p])$ we have as required $(s, n_k) \in coTr(a_k)$ for each $k \in \{1, \ldots, p\}$ and $(s,i) \in coTr(b)$ because $(s,i) \in a \star_d b$. For the third part, we have to prove that the depth of s relative to $i+1$ is at most $d-1$, but it is obvious since the depth relative to i is at most d and $s_{i+1} \to s_i$.

- Otherwise, we have $s_{i+1} \to s_{n_j}$ for $j \in \{1, \ldots, p\}$. Then, we claim that $(s, i+1) \in a_j \star_{d_i - 1} (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p, \{d\}b]$. We do have $(s, i+1) \in Tr(a_j)$ because $(s,i) \in Tr(n[\{d_1\}a_1, \ldots, \{d_p\}a_p])$, thus $(s, n_j) \in coTr(a_j)$ and $(s, i+1) \in Tr(a_j)$ by Lemma 7. It remains to show that $(s, i+1) \in coTr((n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p, \{d\}b])$ and that the depth of s relative to $i+1$ is at most $d_1 - 1$, but the proofs are exactly the same as in the previous case.

$\square$

B Upper bound

Lemma 8 (Monotonicity)**.** *If* $\vdash^{\alpha}_{\rho} a$*, then* $\vdash^{\alpha'}_{\rho'} a$ *for all* $\alpha \leq \alpha'$ *and* $\rho \leq \rho'$*.*

Proof. By induction on a. $\square$

Lemma 9 (Null substitution lemma)**.** *If* $\vdash^{\alpha}_{\rho} a(\emptyset)$ *and the type of* x *in* a *is* 0*, then for all agent* b *we still have* $\vdash^{\alpha}_{\rho} a(b)$*. Moreover, the witness includes as many* Cut *rules as for* $\vdash^{\alpha}_{\rho} a(\emptyset)$*.*

Proof. We prove by induction on the tree witness for $\vdash^{\alpha}_{\rho} a(\emptyset)$ that the above property is true for all context-arena a' such that $a(\emptyset) = a'(\emptyset)$.

- Base. The root of a is 0, hence the result is trivial.
- Red. Suppose a' has the form $n[\{d_1\}a_1, \ldots, \{d_p\}a_p, \{d\}x]$, where $a_1, \ldots, a_p$ possibly include occurrences of x (the case where x appears as a son of the root encompasses the other). The premises of Red are then that for $1 \leq i \leq p$ such that $d_i \geq 1$, $\vdash^{\alpha-1}_{\rho} a_i(\emptyset) \cdot_{d_i - 1} (n-1)[\{d_1\}a_1(\emptyset), \ldots \{d_p\}a_p(\emptyset)]$ and $\vdash^{\alpha-1}_{\rho} (n-1)[\{d_1\}a_1(\emptyset), \ldots \{d_p\}a_p(\emptyset)]$. The induction hypothesis on these premises give witnesses for the two following properties:

$$\vdash^{\alpha-1}_{\rho} (a_i \cdot_{d_i - 1} (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p, \{d\}x])(b) \tag{1}$$

$$\vdash^{\alpha-1}_{\rho} ((n-1)[\{d_1\}a_1, \ldots \{d_p\}a_p, \{d\}x])(b) \tag{2}$$

  All the possible reductions are already covered since $d = 0$, thus by Red we have $\vdash^{(\alpha-1)+1}_{\rho} a(b)$ as required.

- Cut. Let us suppose $\vdash^{\alpha+\gamma}_{\rho} a(\emptyset)$ is obtained by Cut, hence $a(\emptyset)$ has the form $a_1(\emptyset) \cdot_{d_1} a_2(\emptyset)$. Let us suppose that a' has the form $(a_1 \cdot_{d'} a_2) \cdot_d x$, since once again the case where x is a child of the root of a' encompasses the other. The premises of Cut are then $\vdash^{\alpha}_{\rho} a_1(\emptyset)$ and $\vdash^{\gamma}_{\rho} a_2(\emptyset)$, and $d' \leq \rho$. Note now that we also have $(a_1 \cdot_d x)(\emptyset) = a_1(\emptyset)$, therefore the induction hypothesis on $\vdash^{\alpha}_{\rho} a_1(\emptyset)$ along with $\vdash^{\beta}_{\rho} b$ and $d \leq \rho + 1$ implies that $\vdash^{\alpha}_{\rho} a_1(b) \cdot_d b$. But by induction hypothesis we also have $\vdash^{\gamma}_{\rho} a_2(b)$, hence by Cut:
$$\vdash^{\alpha+\gamma}_{\rho} (a_1(b) \cdot_d b) \cdot_{d'} a_2(b)$$
Which was what was required for $(a_1(b) \cdot_{d'} a_2(b)) \cdot_d b$, thus it suffices since trees are considered up to permutation.

□

Lemma 10 (Main substitution lemma)**.** *If $\vdash^{\alpha}_{\rho} a(\emptyset)$, $\vdash^{\beta}_{\rho} b$ and $d \leq \rho + 1$ (where d is the type of x in a), then $\vdash^{\alpha(\beta+1)}_{\rho} a(b)$*

Proof. We prove by induction on the tree witness for $\vdash^{\alpha}_{\rho} a(\emptyset)$ that the above property is true for all context-arena a' such that $a(\emptyset) = a'(\emptyset)$.

- Base. The root of a is 0, hence the result is trivial.

- Red. Suppose a' has the form $n[\{d_1\}a_1, \ldots, \{d_p\}a_p, \{d\}x]$, where $a_1, \ldots, a_p$ possibly include occurrences of x (the case where x appears as a son of the root encompasses the other). The premises of Red are then that for $1 \leq i \leq p$ such that $d_i \geq 1$, $\vdash^{\alpha-1}_{\rho} a_i(\emptyset) \cdot_{d_i-1} (n-1)[\{d_1\}a_1(\emptyset), \ldots \{d_p\}a_p(\emptyset)]$ and $\vdash^{\alpha-1}_{\rho} (n-1)[\{d_1\}a_1(\emptyset), \ldots \{d_p\}a_p(\emptyset)]$. The induction hypothesis on these premises give witnesses for the two following properties:
$$\vdash^{(\alpha-1)(\beta+1)}_{\rho} (a_i \cdot_{d_i-1} (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p, \{d\}x])(b) \tag{3}$$
$$\vdash^{(\alpha-1)(\beta+1)}_{\rho} ((n-1)[\{d_1\}a_1, \ldots \{d_p\}a_p, \{d\}x])(b) \tag{4}$$
By hypothesis we have $\vdash^{\beta}_{\rho} b$, hence by Cut (since $d-1 \leq \rho$), we have:
$$\vdash^{(\alpha-1)(\beta+1)+\beta}_{\rho} b \cdot_{d-1} (n-1)[\{d_1\}a_1(b), \ldots \{d_p\}a_p(b), \{d\}b] \tag{5}$$
Using (3) for all $i \in \{1, \ldots, p\}$, (4) (adjusted to $\vdash^{(\alpha-1)(\beta+1)+\beta}_{\rho}$ by Lemma 8) and (5) we deduce by Red that
$$\vdash^{(\alpha-1)(\beta+1)+\beta+1}_{\rho} n[\{d_1\}a_1(b), \ldots, \{d_p\}a_p(b), \{d\}b]$$
Which is what was required.

- Cut. Let us suppose $\vdash^{\alpha+\gamma}_{\rho} a(\emptyset)$ is obtained by Cut, hence $a(\emptyset)$ has the form $a_1(\emptyset) \cdot_{d_1} a_2(\emptyset)$. Let us suppose that a' has the form $(a_1 \cdot_{d'} a_2) \cdot_d x$, since once again the case where x is a child of the root of a' encompasses the other. The premises of Cut are then $\vdash^{\alpha}_{\rho} a_1(\emptyset)$ and $\vdash^{\gamma}_{\rho} a_2(\emptyset)$, and $d' \leq \rho$. Note now that we also have $(a_1 \cdot_d x)(\emptyset) = a_1(\emptyset)$, therefore the induction hypothesis on $\vdash^{\alpha}_{\rho} a_1(\emptyset)$ along with $\vdash^{\beta}_{\rho} b$ and $d \leq \rho+1$ implies that $\vdash^{\alpha(\beta+1)}_{\rho} a_1(b) \cdot_d b$. But by induction hypothesis we also have $\vdash^{\gamma(\beta+1)}_{\rho} a_2(b)$, hence by Cut:
$$\vdash^{\alpha(\beta+1)+\gamma(\beta+1)}_{\rho} (a_1(b) \cdot_d b) \cdot_{d'} a_2(b)$$
Which was what was required for $(a_1(b) \cdot_{d'} a_2(b)) \cdot_d b$, thus it suffices since trees are considered up to permutation.

□

Lemma 11 (Cut elimination lemma)**.** *Suppose* $\vdash^{\alpha}_{\rho+1} a$. *Then if* $\alpha = 0$, $\vdash^{0}_{\rho} a$. *Otherwise,* $\vdash^{2^{\alpha-1}}_{\rho} a$.

Proof. By induction on the tree witness for $\vdash^{\alpha}_{\rho+1} a$.

- Base. Trivial.

- Red. Suppose $a = n[\{d_1\}a_1, \ldots, \{d_p\}a_p]$, the premises of Red are $\vdash^{\alpha-1}_{\rho+1} a_i \cdot_{d_i-1} (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p]$ for all $i \in \{1, \ldots, p\}$ and $\vdash^{\alpha-1}_{\rho+1} (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p]$. If $\alpha \geq 2$, then it follows by induction hypothesis that $\vdash^{2^{\alpha-2}}_{\rho} a_i \cdot_{d_i-1} (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p]$ and $\vdash^{2^{\alpha-2}}_{\rho} (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p]$, which implies by Red and Lemma 8 that $\vdash^{2^{\alpha-1}}_{\rho} a$. If $\alpha = 1$, then the premises of Red are $\vdash^{0}_{\rho+1} a_i \cdot_{d_i-1} (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p]$ for all $i \in \{1, \ldots, p\}$ and $\vdash^{0}_{\rho+1} (n-1)[\{d_1\}a_1, \ldots, \{d_p\}a_p]$. By induction hypothesis this is still true with ρ instead of $\rho+1$, thus by Red we have $\vdash^{1}_{\rho} [\{d_1\}a_1, \ldots, \{d_p\}a_p]$ which is what we needed to prove.

- Cut. Suppose $a = a_1 \cdot_d a_2$, the premises of Cut are $\vdash^{\alpha}_{\rho+1} a_1$, $\vdash^{\beta}_{\rho+1} a_2$ and $d \leq \rho + 1$. If $\alpha, \beta \geq 1$ then by induction hypothesis it follows that $\vdash^{2^{\alpha-1}}_{\rho} a_1$ and $\vdash^{2^{\beta-1}}_{\rho} a_2$, in particular if we define a context-agent $a'_1 = a_1 \cdot_d x$ we have $\vdash^{2^{\alpha-1}}_{\rho} a'_1(\emptyset)$, hence by the substitution lemma (since $d \leq \rho+1$) we have $\vdash^{2^{\alpha-1}(2^{\beta-1}+1)}_{\rho} a'_1(a_2) = a_1 \cdot_d a_2 = a$, thus $\vdash^{2^{\alpha+\beta-1}}_{\rho} a$ thanks to Lemma 8 (since it is always true than $2^{\alpha+\beta-1} \geq 2^{\alpha-1}(2^{\beta-1}+1)$). If $\alpha = 0$ then by induction hypothesis we have $\vdash^{0}_{\rho} a_1$ and $\vdash^{\beta'}_{\rho} a_2$. We use then the substitution lemma (since $d \leq \rho+1$) to get $\vdash^{0}_{\rho} (a_1 \cdot_d a_2)$, which is stronger that what was required whatever was the value of β. The last remaining case is when $\alpha = 1$ and $\beta = 0$, then by induction hypothesis $\vdash^{1}_{\rho} a_1$ and $\vdash^{0}_{\rho} a_1$, thus by the substitution lemma we have as required $\vdash^{1}_{\rho} (a_1 \cdot_d a_2)$.

□

Lemma 12 (Recomposition lemma)**.** *Let a be an agent. Then:*

$$\vdash^{max(a)|a|}_{depth(a)} a$$

Where $depth(a)$ is the maximal label of an edge in a, $max(a)$ is the maximal label of a node and $|a|$ is the number of nodes.

Proof. First, let us show that the following rule BASE' is admissible, for any α and ρ.

$$\vdash^{\alpha+n}_{\rho} n[]$$

If $n = 0$ this is exactly BASE. Otherwise we apply RED. There is no possible reduction, so the only thing we have to prove is $\vdash^{\alpha+n-1}_{\rho} (n-1)[]$, which is provided by the induction hypothesis. Then we prove the lemma by immediate induction on a, using only BASE', CUT and Lemma 8. □

From now on, let $N(a)$ denote the longest reduction sequence of a. We also use the notations $2^n_0 = n$ and $2^n_{d+1} = 2^{2^n_d}$ for iterated exponentials.

Lemma 13 (Bound lemma)**.** *Let a be an agent, then if $\vdash^{\alpha}_{0} a$, $N(a) \leq \alpha$.*

Proof. First of all we prove that if there is a witness for $\vdash^{\alpha}_{0} a$, then it can be supposed CUT-free: this is proved by induction on $\vdash^{\alpha}_{0} a$, eliminating each use of CUT by Lemma 9. Then, by induction on the CUT-free witness tree for $\vdash^{\alpha}_{0} a$:

- BASE. Then, the root of a is 0, thus $N(a) = 0$; there is nothing to prove.
- RED. The premises of $\vdash^{\alpha}_{0} a$ include in particular that for all a' such that $a \rightsquigarrow a'$, we have $\vdash^{\alpha-1}_{0} a'$. By induction hypothesis, this means that for all such a' we have $N(a') \leq \alpha - 1$, hence $N(a) \leq \alpha$.

□

From all this, it is possible to give a first upper bound by using the recomposition lemma, then iterating the cut elimination lemma. However, we will first prove here a refined version of the cut elimination lemma when $\rho = 1$, which will allow to decrease by one the height of the tower of exponentials. First, we need the following adaptation of the substitution lemma:

Lemma 14 (Base substitution lemma)**.** *If $\vdash^{\alpha}_{0} a(\emptyset)$, $\vdash^{\beta}_{0} b$ and the type of x in a is 1, then $\vdash^{\alpha+\beta}_{0} a(b)$.*

Proof. We prove by induction on the tree witness for $\vdash^{\alpha}_{0} a(\emptyset)$ that the above property is true for all context-arena a' such that $a(\emptyset) = a'(\emptyset)$.

- BASE. The root of a is 0, hence the result is trivial.

- RED. Suppose a' has the form $n[\{d_1\}a_1, \dots, \{d_p\}a_p, \{d\}x]$, where $a_1, \dots, a_p$ possibly include occurrences of x (the case where x appears as a son of the root encompasses the other). The premises of RED are then that for $1 \le i \le p$ such that $d_i \ge 1$, $\mid\!\frac{\alpha-1}{0}\; a_i(\emptyset) \cdot_{d_i-1} (n-1)[\{d_1\}a_1(\emptyset), \dots \{d_p\}a_p(\emptyset)]$ and $\mid\!\frac{\alpha-1}{0}\; (n-1)[\{d_1\}a_1(\emptyset), \dots \{d_p\}a_p(\emptyset)]$. The induction hypothesis on these premises give witnesses for the two following properties:

$$\mid\!\frac{\alpha-1+\beta}{0}\; (a_i \cdot_{d_i-1} (n-1)[\{d_1\}a_1, \dots, \{d_p\}a_p, \{d\}x])(b) \tag{6}$$

$$\mid\!\frac{\alpha-1+\beta}{0}\; ((n-1)[\{d_1\}a_1, \dots \{d_p\}a_p, \{d\}x])(b) \tag{7}$$

  By hypothesis we have $\mid\!\frac{\beta}{0}\; b$, hence by Lemma 9 (since $d = 1$) we have

$$\mid\!\frac{\beta}{0}\; b \cdot_{d-1} (n-1)[\{d_1\}a_1(b), \dots \{d_p\}a_p(b), \{d\}b]$$

  Hence, using (6) for all $i \in \{1, \dots, p\}$, (7) and (B) (adjusted to $\mid\!\frac{\alpha-1+\beta}{0}\; b$ by Lemma 8) we deduce by RED that

$$\mid\!\frac{\alpha+\beta}{0}\; n[\{d_1\}a_1(b), \dots, \{d_p\}a_p(b), \{d\}b]$$

  Which is what was required.

- CUT. Let us suppose $\mid\!\frac{\alpha+\gamma}{0}\; a(\emptyset)$ is obtained by CUT, hence $a(\emptyset)$ has the form $a_1(\emptyset) \cdot_0 a_2(\emptyset)$. Let us suppose that a' has the form $(a_1 \cdot_0 a_2) \cdot_d x$. The premises of CUT are then $\mid\!\frac{\alpha}{0}\; a_1(\emptyset)$ and $\mid\!\frac{\gamma}{0}\; a_2(\emptyset)$. Note now that we also have $(a_1 \cdot_d x)(\emptyset) = a_1(\emptyset)$, therefore the induction hypothesis on $\mid\!\frac{\alpha}{\rho}\; a_1(\emptyset)$ along with $\mid\!\frac{\beta}{0}\; b$ implies that $\mid\!\frac{\alpha+\beta}{0}\; a_1(b) \cdot_1 b$ and all that remains is to substitute $a_2(b)$ in $(a_1(b) \cdot_1 b) \cdot_0 x$. But since the type of x is 0, Lemma 9 proves that $\mid\!\frac{\alpha+\beta}{0}\; (a_1(b) \cdot_d b) \cdot_{d'} a_2(b)$, which concludes since trees are considered up to permutation.

□

Lemma 15 (Base cut elimination lemma)**.** *If* $\mid\!\frac{\alpha}{1}\; a$, *then* $\mid\!\frac{\alpha}{0}\; a$.

Proof. By induction on the witness tree for $\mid\!\frac{\alpha}{1}\; a$.

- BASE. Trivial.

- RED. Suppose a has the form $n[\{d_1\}a_1, \dots, \{d_p\}a_p]$. The premises of RED are that for all $i \in \{1, \dots, p\}$ we have $\mid\!\frac{\alpha-1}{1}\; a_1 \cdot_{d_i-1} (n-1)\{d_1\}a_1, \dots, \{d_p\}a_p]$ and $\mid\!\frac{\alpha-1}{1}\; (n-1)\{d_1\}a_1, \dots, \{d_p\}a_p]$. The result is then trivial by induction hypothesis and RED.

- Cut. Suppose $a = a_1 \cdot_d a_2$ with $d \leq 1$, the premises of Cut are that $\vdash_1^{\alpha} a_1$ and $\vdash_1^{\beta} a_2$. If $d = 0$, then the result is trivial by the induction hypothesis and Cut. If $d = 1$, we just apply Lemma 14 instead of Cut.

$\square$

Theorem 2 (Upper bound)**.** *Let $depth(a)$ denote the highest edge label in a, $max(a)$ means the highest node label and $|a|$ means the number of nodes of a. Then if $depth(a) \geq 1$ and $max(a) \geq 1$ we have:*

$$N(a) \leq 2^{max(a)|a|-1}_{depth(a)-1}$$

For the particular case when $a = n[\{d\}p[]]$ and if $d \geq 2$ we have:

$$N_d(n,p) \leq 2^{n(p+1)}_{d-2}$$

Proof. Let us first prove the first part. By the recomposition lemma, we have $\vdash_{depth(a)}^{max(a)|a|} a$. By $depth(a) - 1$ iterations of the cut elimination lemma, we have $\vdash_{1}^{2^{max(a)|a|-1}_{depth(a)-1}} a$. But then by Lemma 15 we also have $\vdash_{0}^{2^{max(a)|a|-1}_{depth(a)-1}} a$. By Lemma 13, this implies as required that $N(a) \leq 2^{max(a)|a|-1}_{depth(a)-1}$. We turn now to the second part. Obviously, we have $\vdash_{d-1}^{n} n[]$ and $\vdash_{d-1}^{p} p[]$. By the substitution lemma, this implies that $\vdash_{d-1}^{n(p+1)} n[\{d\}p[]]$. By $d-2$ applications of the cut elimination lemma, and one application of Lemma 15, this means that $\vdash_{0}^{2^{n(p+1)-1}_{d-1}} [\{d\}p[]]$ hence $N_d(n,p) \leq 2^{n(p+1)-1}_{d-2}$. $\square$