

THE FINE STRUCTURE OF LD-EQUIVALENCE

PATRICK DEHORNOY

Abstract. We introduce new algebraic techniques for the study of left self-distributivity. We establish a self-similarity propriety for the terms $\partial^k t$ which are counterparts to Garside's fundamental braids Δ_n^k , and deduce partial answers to several long-standing open questions: convergence of the Polish Algorithm, computation of the normal form, existence of a lattice structure on LD-equivalence classes.

Key words: self-distributivity, braid groups, word problem

AMS Subject Classification: 20N02, 20F36.

The left self-distributivity identity

$$x(yz) = (xy)(xz). \quad LD)$$

has received some attention in recent years because of its connection with the theory of braids and knots on the one hand [10], and with axiomatic set theory in the other hand [21], [24]. Define an LD-system to be a set equipped with a binary operation that satisfies (LD) . The existence of a canonical linear ordering on free LD-systems has led to the construction of a left invariant linear ordering on Artin's braid group B_∞ , and, from there, to a number of new results about braid groups [12], [23], [1], [18], Artin groups [14], and mapping class groups [25]. The goal of this paper is to use the notions of injective, steep, and perfect terms (as introduced in Sections 1 and 2) to get partial results about three of major problems that are left open in free LD-systems: the convergence of the Polish Algorithm, the computation of the right normal form, and the Embedding Conjecture. These mutually independent applications are treated in Sections 3, 4, and 5 respectively.

Let us briefly describe the framework, so as to be able to state the results precisely.

Assume that X is a nonempty set. By standard arguments, the free LD-system based on X is the quotient of the absolutely free system T_X consisting of all well formed terms with variables in X under the least congruence on T_X that forces the identity (LD) to hold, *i.e.*, the congruence $=_{LD}$ generated by all pairs of the form $(t_1(t_2t_3), (t_1t_2)(t_1t_3))$. We say that two terms t, t' are *LD-equivalent* when $t =_{LD} t'$ holds, *i.e.*, when t can be transformed to t' using Identity (LD) . LD-equivalence is a very complex relation. The above mentioned results about braids originate in its properties, which have been investigated in [2], [6], [7], [21], [22], [23], among other references. Many questions about LD-equivalence remain open, and, in particular, the structure of the LD-equivalence class of a given term is not completely known—in contradistinction to the syntactically close case of the associativity identity $x(yz) = (xy)z$, where the equivalence class of a term is the finite set of all possible bracketings of the corresponding list of variables. Let us say that the term t' is an LD-expansion of the term t if t can be transformed into t' by iteratively replacing subterms of the form $t_1(t_2t_3)$ with the corresponding term $(t_1t_2)(t_1t_3)$. It is known that two terms are LD-equivalent if and only if they admit a common LD-expansion, a property directly connected with the fact that the braid ordering of [6] is a linear ordering. The proof relies

upon the existence, for every term t , of a distinguished LD-expansion ∂t that simultaneously expands all LD-expansions obtained using (LD) once exactly. Thus ∂t is a sort of lower common LD-expansion of all basic LD-expansions of t . The role of ∂ for the study of (LD) is reminiscent of the role of Garside's fundamental elements Δ_n in the study of braids [19]. The main point here is that the sequence $t, \partial t, \partial^2 t, \dots$ is cofinal in the LD-equivalence class of t with respect to the relation of being an LD-expansion, and, to a large extent, understanding how LD-classes are made means controlling the properties of the terms $\partial^k t$. Let us say that a term t is injective if no variable occurs twice in t . Our main technical result here is

Proposition A. *Assume that t is an injective term. Then, for every k , the term $\partial^k t$ is perfect.*

Perfectness is a geometrical property: roughly speaking, a term t is perfect when it satisfies the self-similarity condition that every geometrical pattern appearing in the associated tree at some position α then reappears at every position on the right of α . The result of Proposition A (in a different form) has been known in the particular case $k = 1$ for more than ten years [2], [3], but, due to the complexity of the terms $\partial^k t$ for $k \geq 2$, the previous attempts to prove the general case had failed [7], [11].

We use Proposition A to prove new partial results about three open questions involving left self-distributivity. In doing so, we prove the conjectures of [3] and [4], and achieve the program of [7] and [11] in the sense that a completely new approach will presumably be necessary to solve those questions that remain open.

The first application deals with the Polish Algorithm. The word problem of LD-equivalence, *i.e.*, the question of recognizing whether two given terms are LD-equivalent or not, is known to be solvable [6]: when we are given two terms t_1, t_2 involving only one variable x , we can decide whether $t_1 =_{LD} t_2$ holds by evaluating t_1 and t_2 in Artin's braid group B_∞ by mapping x to the unit braid 1 and using on B_∞ the braid exponentiation $b_1 \wedge b_2 = b_1 \text{sh}(b_2) \sigma_1 \text{sh}(b_1^{-1})$, where sh is the shift endomorphism that maps σ_i to σ_{i+1} for every i ; the case of terms with more than one variable can be solved similarly using an extension of the braid group B_∞ . Besides such semantical algorithms, there exists a simple, natural syntactic algorithm, which compares the terms t_1 and t_2 by looking at their right Polish expressions. If they are equal, the terms are equal, hence LD-equivalent. Otherwise, we look at the leftmost clash between the Polish expressions. If the clash involves different variables, or if one word is a prefix of the other, then t_1 and t_2 are LD-unequivalent. Otherwise, there exists a canonical way to solve the clash by expanding one of the terms using left self-distributivity. The Polish Algorithm consists in iterating the process until the decision can be made [11]. Despite the efforts of several researchers, the convergence question is still open: it is not known whether the algorithm always comes to an end in a finite number of steps. Here we prove the following result:

Proposition B. *Assume that t_1 and t_2 are LD-equivalent to some injective term, or that t_1 and t_2 are LD-expansions of a common third term. Then the Polish Algorithm running on (t_1, t_2) converges in a finite number of steps.*

This implies in particular that the Polish algorithm running on every pair of the form $(t, \partial^k t)$ always converges, as was conjectured in [3].

The next application deals with the computation of the right normal form. The question is to effectively select a distinguished element in every LD-equivalence class. R. Laver has constructed solutions in [21] and [22]. No complexity bound is known for the computation of these normal forms, sometimes referred to as left normal. Alternatively, right normal forms have been defined in [7] using the terms $\partial^k t$, and a primitive recursive upper bound exists for its computation. However, computing the normal form of an arbitrary term is a

very complicated task, for which presumably no general tractable method exists. However, the right normal form involves an additional parameter, namely a term t_0 that is used as a fixed reference term (like a basis in the expansion of the integers or of the rationals). Here we give a renewed approach by introducing the notion of a fractional cut of a term, which hopefully makes the whole construction more understandable. Then, we introduce the auxiliary notion of the table associated with a term, and, using Proposition A, we prove:

Proposition C. *Assume that t_0 is an injective term. Then there exists a tractable algorithm that computes the table of t_0 .*

We then show how to possibly use the table of t_0 to compute the t_0 -normal form of a term t , or, more generally, its t'_0 -normal form for every substitute t'_0 of t_0 of a convenient type. The new construction does not work for every term t , but the results could nevertheless be optimal in some sense, as computing the normal form in the most general case seems to be intrinsically intractable.

Finally, we consider the Embedding Conjecture. This statement admits several equivalent forms. One of them asserts that, for every term t , the LD-equivalence class of t equipped with the LD-expansion relation is an upper semilattice, *i.e.*, every two LD-equivalent terms admit a lower common LD-expansion. Other forms involve a certain group G_{LD} that describes the geometry of left self-distributivity [6], [9], and an associated monoid M_{LD} . The connection between braids and left self-distributivity originates in the group G_{LD} being an extension of Artin's braid group B_∞ , and, in particular, the monoid M_{LD} is an extension of the monoid B_∞^+ of positive braids. The Embedding Conjecture claims that the monoid M_{LD} embeds in the group G_{LD} , and it is a refinement of the well known result by Garside that the monoid B_∞^+ of positive braids embeds in the group B_∞ [19]. Let us say that the Embedding Conjecture is true for some element a of M_{LD} if the canonical homomorphism f of M_{LD} into G_{LD} is injective at a , *i.e.*, if $b \neq a$ implies $f(b) \neq f(a)$. For every term t and every nonnegative integer k , there exists an element $\Delta_t^{(k)}$ of M_{LD} that describes how $\partial^k t$ is obtained from t , and the elements $\Delta_t^{(k)}$ are exact counterparts for the braids Δ_n^k in B_∞^+ . Here we prove:

Proposition D. *The Embedding Conjecture is true for every element of M_{LD} that is a right divisor of some element of the form $\Delta_t^{(k)}$.*

This result—which, by projection, implies Garside's embedding result for braids—implies in particular that, for each term t , and each nonnegative integer k , the subset of the LD-equivalence class of t consisting of those LD-expansions of t of which $\partial^k t$ is an LD-expansion is a lattice.

The paper is organized as follows. In Section 1, we introduce the vocabulary and recall earlier results about LD-equivalence. In Section 2, we introduce the notion of a perfect term, and, building on the results of [7] about the geometry of the terms ∂t and of [6] about orders in free LD-systems, we investigate the so-called covering relation of the term ∂t , and establish Proposition A. In Section 3, we apply the results to the Polish Algorithm, and deduce in particular Proposition B. Section 4 is devoted to normal forms and Proposition C. Finally, the Embedding Conjecture is discussed in Section 5, where Proposition D and further results are proved using the notion of a confluent family of terms. There is (almost) no interaction between the latter three sections.

All notions used in the current paper are defined precisely, and all earlier results needed in the proofs are mentioned. However, some knowledge of [5], [6], and, mainly, [7] is certainly helpful for Sections 2 and 4, even if not formally needed. Section 5 uses techniques developed in [8] and [14].

1. PRELIMINARIES

We recall in this introductory section definitions and earlier results about left self-distributivity needed in the sequel, in particular the notion of an LD-expansion of a term, the operation ∂ on terms, and the notions of a cut and of a descent of a term.

LD-equivalence and LD-expansions

Everwhere in the paper, we use the convention that missing brackets in algebraic expressions are to be added on the right: abc stands for $a(bc)$.

In the sequel, we fix an infinite sequence of variables x_1, x_2, \dots , and denote by T_∞ the set of all well formed terms constructed using these variables and a binary operation symbol (usually skipped here). Thus x_1 and $x_2(x_1x_3)$ are typical elements of T_∞ . We use T_1 for the set of those terms involving the variable x_1 only, and write FLD_∞ for T_∞/\equiv_{LD} , and FLD_1 for T_1/\equiv_{LD} . By construction, FLD_∞ is the free LD-system based on $\{x_1, x_2, \dots\}$, while FLD_1 is the free LD-system based on $\{x_1\}$. For t a term in T_∞ , the LD-equivalence class of t is denoted by \bar{t} ; the size of t is defined to be the number of occurrences of variables in t .

As geometric features are crucial here, it is convenient to associate with every term a finite binary tree whose leaves are labelled with variables: if t is the variable x , the tree associated with t consists of a single node labelled x , while, for $t = t_1t_2$, the binary tree associated with t has a root with two immediate successors, namely a left one which is (the tree associated with) t_1 , and a right one which is (the tree associated with) t_2 . For instance, the tree associated with the term $x_2(x_1x_3)$ is $x_2 \begin{array}{c} \diagup \quad \diagdown \\ x_1 \quad x_3 \end{array}$. We use finite sequences

of 0's and 1's as addresses for the nodes in such trees, starting with an empty address ϕ for the root, and using 0 and 1 for going to the left and to the right respectively. For t a term, we define the *outline* of t to be the collection of all addresses of leaves in (the tree associated with) t , and the *skeleton* of t to be the collection of the addresses of nodes in t : thus, for instance, the outline of the term $x_2(x_1x_3)$ is the set $\{0, 10, 11\}$, while its skeleton is $\{0, 10, 11, 1, \phi\}$, as t comprises three leaves and two internal nodes.

For t a term, and α an address in the skeleton of t , we have the natural notion of the α -th subterm of t , denoted $\text{sub}(t, \alpha)$: this is the term corresponding to the subtree of the tree associated with t whose root lies at address α . This amounts to defining inductively

$$\text{sub}(t, \alpha) = \begin{cases} t & \text{if } t \text{ is a variable or } \alpha = \phi \text{ holds,} \\ \text{sub}(t_0, \beta) & \text{for } t = t_0t_1 \text{ and } \alpha = 0\beta, \\ \text{sub}(t_1, \beta) & \text{for } t = t_0t_1 \text{ and } \alpha = 1\beta. \end{cases}$$

If the address α belongs to the outline of the term t , the subterm $\text{sub}(t, \alpha)$ consists of a single variable, and we shall denote it by $\text{var}(t, \alpha)$.

With addresses at hand, we can introduce a local version of LD-expansion:

Definition. Assume that t is a term, and α is an address such that $\alpha 10$ belongs to the outline of t . Then we define $(t)\alpha$ to be the LD-expansion of t obtained by replacing the subterm $\text{sub}(t, \alpha)$ with the term $(\text{sub}(t, \alpha 0)\text{sub}(t, \alpha 10))(\text{sub}(t, \alpha 0)\text{sub}(t, \alpha 11))$.

Thus $(t)\alpha$ is the term obtained from t by applying left self-distributivity *at* α in the expanding direction. We say that t' is a k -LD-expansion of t if there exists a finite sequence of k addresses $\alpha_1, \dots, \alpha_k$ satisfying $t' = (\dots((t)\alpha_1)\alpha_2)\dots)\alpha_k$ — also denoted $(t)\alpha_1 \cdots \alpha_k$. By definition, a term t' is an LD-expansion of the term t if and only if it is a k -LD-expansion for some k .

Among all possible LD-expansions of a given term t , a distinguished one plays a crucial role in the sequel, namely the term ∂t , inductively defined as follows: we have $\partial t = t$ for t

a variable, and $\partial t = \partial t_1 * \partial t_2$ for $t = t_1 t_2$, where $*$ is the operation defined by the inductive rules $t_0 * t = t_0 t$ for t a variable, and $t_0 * t = (t_0 * t_1)(t_0 * t_2)$ for $t = t_1 t_2$.

The following statements gather those basic properties of LD-expansions we need in the sequel. The first is proved using induction on k , and the only slightly delicate point is to prove that the operation ∂ is increasing with respect to LD-expansion, *i.e.*, that, if t' is an LD-expansion of t , then $\partial t'$ is an LD-expansion of ∂t . The second result then easily follows. The last result is proved by a direct, straightforward induction.

Proposition 1.1. [2] *If t is a term and t' is a k -LD-expansion of t , then $\partial^k t$ is an LD-expansion of t . In particular, ∂t is an LD-expansion of all 1-LD-expansion of t .*

Proposition 1.2. [2] *Two terms are LD-equivalent if and only if they admit a common LD-expansion. More precisely, t and t' are LD-equivalent if and only if $\partial^k t$ is an LD-expansion of t' for k large enough.*

Proposition 1.3. [6] *Assume that t' is an LD-expansion of t , and $\text{sub}(t, 0^p)$ is defined. Then there exists $p' \geq p$ such that $\text{sub}(t', 0^{p'})$ is an LD-expansion of $\text{sub}(t, 0^p)$.*

Order on free LD-systems

The next ingredients are orders.

Definition. Assume that t_1, t_2 are terms. We say that t_2 is a proper iterated subterm of t_1 , denoted $t_1 \sqsupset t_2$, if we have $t_2 = \text{sub}(t_1, 0^p)$ for some positive p . We say that $t_1 \sqsupset_{LD} t_2$ holds if there exists two terms t'_1, t'_2 satisfying $t'_1 =_{LD} t_1$, $t'_2 =_{LD} t_2$ and $t'_1 \sqsupset t'_2$. Finally, we denote by \sqsupset the relation induced by \sqsupset_{LD} on FLD_∞ .

By definition, $\text{sub}(t_2, 0)$ is the left subterm of t , and, similarly, $\text{sub}(t_2, 0^p)$ is the p -th iterated left subterm of t_2 . So the relation \sqsupset_{LD} on T_∞ is the LD-closure of the relation of being an iterated left subterm. We have the following deep result about left self-distributivity:

Proposition 1.4. [6] *The relation \sqsupset is a partial ordering on FLD_∞ which is compatible with product on the left; its restriction to FLD_1 is a linear ordering.*

Thus, the relation \sqsupset_{LD} is linear on one variable terms in the sense that, for any two terms t_1, t_2 in T_1 , exactly one of $t_1 \sqsupset_{LD} t_2$, $t_1 =_{LD} t_2$, $t_1 \sqsupset_{LD} t_2$ holds. For terms with more than one variable, the result is no longer true. However, using an ordering on variables, we can define a lexicographical extension of \sqsupset_{LD} that induces a linear ordering on FLD_∞ . In order to make the definitions precise, and for future use in Section 3, we introduce the right Polish expression of terms.

Definition. For t a term in T_∞ , the *right Polish expression* of t is defined to be the word \tilde{t} over the alphabet $\{x_1, x_2, \dots, \bullet\}$ inductively defined by the rules: $\tilde{t} = t$ for t a variable, $\tilde{t} = \tilde{t}_1 \tilde{t}_2 \bullet$ for $t = t_1 t_2$.

For instance, if t is the term $x_2(x_1 x_3)$, \tilde{t} is the length 5 word $x_2 x_1 x_3 \bullet \bullet$.

Definition. Assume that t_1, t_2 are terms in T_∞ . We say that $t_1 \gg t_2$ holds if there exist a word w such that \tilde{t}_1 begins with $w x_{i_1}$ while \tilde{t}_2 begins with $w x_{i_2}$ with $i_1 > i_2$. We say that $t_1 \gg_{LD} t_2$ if there exist terms t'_1, t'_2 satisfying $t'_1 =_{LD} t_1$, $t'_2 =_{LD} t_2$, and $t'_1 \gg t'_2$. Finally, we say that $t_1 >_{LD} t_2$ holds if $t_1 \sqsupset_{LD} t_2$ or $t_1 \gg_{LD} t_2$ holds.

Once Proposition 1.4 is established, it is rather easy to deduce:

Proposition 1.5. [5] (i) *The relations \sqsupset_{LD} and \gg_{LD} exclude each other, and the relation $>_{LD}$ induces a linear order on FLD_∞ .*

(ii) *The conjunction of $t_1 \gg_{LD} t_2$, $t'_1 \sqsupset_{LD} t_1$ and $t'_2 \sqsupset_{LD} t_2$ implies $t'_1 \gg_{LD} t'_2$.*

We shall denote by \gg and $>$ respectively the partial order and the linear order on FLD_∞ induced by the relations \gg_{LD} and $>_{LD}$.

The cuts of a term

Besides the subterms of a term, its cuts play a significant role.

Definition. Assume that t is a term in T_∞ , and α is an address in the skeleton of t . The *cut* of t at α is the term $\text{cut}(t, \alpha)$ inductively defined by

$$\text{cut}(t, \alpha) = \begin{cases} t & \text{for } \alpha = \phi, \\ \text{cut}(t_0, \beta) & \text{for } t = t_0 t_1 \text{ and } \alpha = 0\beta, \\ t_0 \text{cut}(t_1, \beta) & \text{for } t = t_0 t_1 \text{ and } \alpha = 1\beta. \end{cases}$$

Addresses are equipped with a natural left-right ordering: we say that the address α lies *on the right* of the address β , denoted $\alpha >_{LR} \beta$, if there exists an address γ such that $\gamma 1$ is, as a word on $\{0, 1\}$, a prefix of α and $\gamma 0$ is a prefix of β . Then, the cut of the term t at α is the term obtained from t by deleting the part of t that lies on the right of α . For instance, the term $\text{cut}(x_2(x_1 x_3), 10)$ is obtained from $x_2(x_1 x_3)$ by removing the part that lies on the right of x_1 : there remains the term $x_2 x_1$. Observe that $\text{cut}(t, \alpha 1^q) = \text{cut}(t, \alpha)$ always holds (provided $\alpha 1^q$ belongs to the skeleton of t). The following computational formula is easy.

Lemma 1.6. [7] *For t a term and α an address in the skeleton of t , we have*

$$\text{cut}(t, \alpha) = \text{sub}(t, \alpha_1 0) \cdots \text{sub}(t, \alpha_p 0) \text{sub}(t, \alpha), \quad (1.1)$$

$$\text{cut}(t, \alpha) =_{LD} \text{cut}(t, \alpha_p 0) \cdots \text{cut}(t, \alpha_1 0) \text{sub}(t, \alpha), \quad (1.2)$$

where $\alpha_1, \dots, \alpha_p$ are those prefixes of α such that $\alpha_1 1, \dots, \alpha_p 1$ are prefixes of α , enumerated with increasing lengths.

These explicit formulas easily provide us with an isomorphism between the left-right ordering of addresses and the \sqsupset_{LD} -ordering of the associated cuts:

Lemma 1.7. [7] *Assume that t is a term in T_∞ , and α, β belong to the outline of t . Then $\text{cut}(t, \alpha) \sqsupset \text{cut}(t, \beta)$ is equivalent to $\alpha >_{LR} \beta$.*

Besides the cuts themselves, we shall also use the LD-equivalence classes of cuts, and consider, for every term t , the family of all LD-equivalence classes of cuts of t .

Definition. For a in FLD_∞ , t in T_∞ , and α in the outline of t , we say that a *appears* in t at α if a is the LD-equivalence class of the term $\text{cut}(t, \alpha)$. The *content* of the term t is defined to be the set of all elements of FLD_∞ appearing in t at some address not of the form 1^p .

Lemma 1.7 shows that, if a and b appear in t at α and β respectively, then $a \sqsupset b$ is equivalent to $\alpha >_{LR} \beta$. This implies in particular that every element a of FLD_∞ appears at most once in a given term t of T_∞ . So there will be no ambiguity in speaking of the address where a appears in t . We shall say that the element a appears in t *below* α to mean that a appears in t at some address that lies below α , *i.e.*, of the form $\alpha\beta$. Also we shall say that a appears in t at $\alpha 1^*$ to mean that a appears at some address of the form $\alpha 1^p$ with $p \geq 0$.

Cuts behave nicely with respect to LD-expansions. The following result is easy.

Lemma 1.8. [10] *Assume that the term t' is an LD-expansion of the term t . Then every element appearing in t appears in t' as well, hence the content of t is included in the content of t' . More precisely, assume $t' = (t)\alpha$. Then the elements appearing in t' are those elements that appear in t , completed with the elements ab such that a appears at $\alpha 10 1^*$ in t and b appears below $\alpha 0$ in t .*

The descents of a term

The last ingredient we shall resort to is the covering relation between addresses, and the derived notion of a descent, which allows one to describe the geometry of the terms ∂t .

We start with a partition the left-right ordering of addresses into two partial orders.

Definition. Assume that α, β are addresses, and α lies on the right of β . We say that α *covers* β if there exists an address γ and an integer p such that $\alpha = \gamma 1^p$ holds and β lies below $\gamma 0$; we say that α *uncovers* β if $\alpha >_{LR} \beta$ holds, but α does not cover β .

The following result is straightforward:

Lemma 1.9. [7] *The covering and uncovering relations are partial orders on addresses. For every address α and every term t such that α lies in the outline of t , there exists a unique address $\mu_t(\alpha)$ in the outline of t such that, for β on the left of α in the outline of t , β is covered by α if and only if $\alpha >_{LR} \beta \geq_{LR} \mu_t(\alpha)$ holds, and β is uncovered by α if and only if $\mu_t(\alpha) >_{LR} \beta$ holds.*

As the left-right ordering of addresses is a partial ordering and is not well founded, there exists no address $\mu(\alpha)$ such that the addresses uncovered by α are those addresses β that satisfy $\mu(\alpha) >_{LR} \beta$, but this becomes true when we restrict to the outline of a term, and the situation is as illustrated in Figure 1.1.

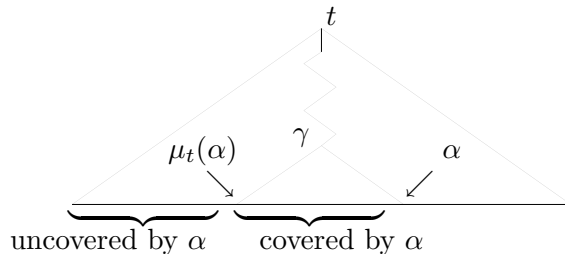


Figure 1.1. The covering relation in t

Definition. Assume that t is a term. A *descent* in t is defined to be a finite sequence of addresses $(\alpha_1, \dots, \alpha_{p+1})$ in the outline of t such that α_i uncovers α_{i+1} for each i .

The interest of considering descents lies in that there exists a bijective correspondence between the descents of a term t and the outline of the term ∂t . The following result is instrumental in the construction of the right normal form. Its proof requires a careful, but natural, inductive argument.

Proposition 1.10. [7] *For every term t , there exists a one-to-one correspondence π_t between the descents in t and the addresses in the outline of ∂t , and, for $\alpha = \pi_t(\alpha_1, \dots, \alpha_{p+1})$, we have*

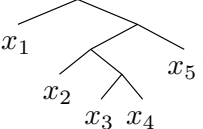
$$\text{cut}(\partial t, \alpha) =_{LD} \text{cut}(t, \alpha_1) \cdots \text{cut}(t, \alpha_{p+1}). \quad (1.3)$$

Moreover, $\pi_t(\alpha_1, \dots, \alpha_{p+1})$ covers $\pi_t(\beta_1, \dots, \beta_{q+1})$ if and only if we have $p \leq q$, $\alpha_1 = \beta_1, \dots, \alpha_p = \beta_p$, and $\alpha_{p+1} > \beta_{p+1}$.

The covering relation of a term

As every element appearing in a term t appears at a well defined address, the covering relation on addresses induces a covering relation for the elements that appear in t .

Definition. Assume that t is a term in T_∞ , and a, b are elements of FLD_∞ . We say that a covers (resp. uncovers) b in t if a and b appear in t , and the address where a appears in t covers (resp. uncovers) the address where b appears in t .

Example 1.11. Let t be the term $x_1(x_2(x_3x_4))x_5$, i.e., . Four elements

appear in t , namely (the LD-classes of) $x_1, x_1x_2, x_1(x_2x_3)$ and $x_1(x_2(x_3x_4))$. In t , x_1x_2 uncovers x_1 , $x_1(x_2x_3)$ uncovers x_1 and x_1x_2 , and $x_1(x_2(x_3x_4))$ uncovers x_1 , but it covers x_1x_2 and $x_1(x_2x_3)$.

The following result expresses the geometrical property that expanding a term t at α causes the cut of t at $\alpha 0$ to cover all cuts of t at addresses below $\alpha 0$. The result easily follows from the explicit values of Lemma 1.6.

Lemma 1.12. [10] (i) *Assume that the term t' is an LD-expansion of the term t , and that a covers b in t . Then a covers b in t' .*

(ii) *More precisely, assume $t' = (t)\alpha$, and let a be the element that appears at $\alpha 0 1^*$ in t . Then the covering graph of t' is the covering graph of t , completed with the following pairs:*

- all pairs (a, b) with b appearing below $\alpha 0$ in t ;
- all pairs (ab, ac) with b, c appearing below $\alpha 0$ in t and b covering c in t ;
- all pairs (c, ab) with b appearing below $\alpha 0$ in t and c covering a in t .

By Lemma 1.7, for every term t , the elements that appear in t make a chain with respect to \sqsubset , and this chain is isomorphic to the outline of t ordered by the left-right ordering. By Lemma 1.9, for every address α in the outline of t , the addresses β that are covered by α are those addresses that satisfy $\alpha >_{LR} \beta \geq_{LR} \mu_t(\alpha)$. Thus, we deduce:

Lemma 1.13. *Assume that t is a term, and a appears at α in t . Then the elements covered by a in t are those elements b that appear in t and satisfy $a \sqsupset b \sqsupseteq a_0$, where a_0 is the element that appears at $\mu_t(\alpha)$ in t .*

Definition. For a, a' elements of FLD_∞ , we say that a and a' are *almost equal* if they can be represented by terms that are LD-equivalent up to a possible change of the rightmost variable.

As the rightmost variable of a term is preserved under left self-distributivity, the previous definition is non-ambiguous. Using Formula (1.2), we obtain:

Lemma 1.14. *Assume that t is a term, and a appears in t at α . Let a_0 be the element that appears in t at $\mu_t(\alpha)$. Then the successor of a in the content of t is almost equal to aa_0 .*

2. THE GEOMETRY OF $\partial^k t$

We enter now the core of the study. The aim of this section is to prove Proposition A, which amounts to describing the geometry of the terms $\partial^k t$. The proof uses an inductive argument, and the main step consists in establishing that two properties of terms called steepness and fullness are preserved under operation ∂ . Steepness is an order property relying on Proposition 1.5, while fullness involves the content and the covering relation, and the argument relies on the description of the cuts of ∂t as given in Proposition 1.10.

Steep terms

We start from the notion of an increasing term, and its weakening, the notion of a quasi-increasing term. We recall that, for t a term, and α an address in the outline of t , $\text{var}(t, \alpha)$ denotes the variable that occurs at α in t . We write $\text{var}(t, \alpha 0^*)$ for $\text{var}(t, \alpha 0^p)$, where p is the unique integer such that $\alpha 0^p$ belongs to the outline of t (which exists if and only if α belongs to the skeleton of t).

Definition. Assume that t is a term in T_∞ . We say that t is *increasing* if the variables of t enumerated from left to right make a strictly increasing sequence. We say that t is *quasi-increasing* if, for every internal address α in the skeleton of t , the inequality $\text{var}(t, \alpha 10^*) > \text{var}(t, \alpha 0^*)$ holds.

By definition, every increasing term is injective, *i.e.*, no variable occurs twice, and it is quasi-increasing, as $\alpha 10^* >_{LR} \alpha 0^*$ holds. The converse implication is not true. For instance, the term $(x_1 x_2) x_2$ is not increasing, but it is quasi-increasing: there are two internal addresses, namely ϕ and 0 , and, in both cases, $\text{var}(t, \alpha 10^*)$ is x_2 , while $\text{var}(t, \alpha 0^*)$ is x_1 .

The following technical result will be crucial.

Lemma 2.1. *Assume that t is a quasi-increasing term, and a covers b in t . Then $\bar{t} \gg ab$ holds, and, therefore, $\bar{t} \sqsupset ab$ does not.*

Proof. Assume that a appears at α in t , and b appears at β . Let α_0 be the least address covered by α in the outline of t , and α^+ be the immediate successor of α in this set. Let a_0 denote the LD-class of $\text{cut}(t, \alpha_0)$, and a^+ denote the LD-class of $\text{cut}(t, \alpha^+)$. By construction, we have $\alpha_0 = \mu_t(\alpha)$, so, by Lemma 1.14, $\text{cut}(t, \alpha^+)$ is LD-equivalent to the term obtained from $\text{cut}(t, \alpha) \text{cut}(t, \alpha_0)$ by replacing the right variable $\text{var}(t, \alpha_0)$ with the variable $\text{var}(t, \alpha^+)$. Let γ be the greatest common prefix of the addresses α and α^+ . Applying the hypothesis that t is quasi-increasing at γ gives $\text{var}(t, \alpha^+) > \text{var}(t, \alpha_0)$, and we deduce $a^+ \gg aa_0$. By construction, we have $\bar{t} \sqsupseteq a^+$, and $b \sqsupseteq a_0$, hence $ab \sqsupseteq aa_0$, so Proposition 1.5(ii) implies $\bar{t} \gg ab$. ■

It is not true that every LD-expansion of a (quasi)increasing term need be (quasi)increasing. We introduce now a new property, weaker than (quasi)increasingness, but preserved under LD-equivalence. For a in FLD_∞ , we denote by $a^{[2]}$ the element aa .

Definition. For a in FLD_∞ , we say that a is *steep* if $a > b$ implies $a \gg b^{[2]}$ for every b . For t in T_∞ , we say that t is *steep* if the LD-class of t is steep.

As $a \gg b$ always implies $a \gg b^{[2]}$, the non-trivial part in the definition involves those elements b that satisfy $a \sqsupset b$. Observe that no element of FLD_1 but the generator x_1 is steep. The following criterion characterizes steep terms by a purely local condition, one that involves t only, and not the LD-expansions of t . It will allow us to construct a number of steep terms.

Proposition 2.2. (i) Assume that t is a term. Then t is steep if and only if the inequality $\bar{t} \gg a^{[2]}$ holds for every element a appearing in t .

(ii) Assume $t = t_1 t_2$. A sufficient condition for t to be steep is that t_1 and t_2 are steep, and $t_2 \gg_{LD} t_1$ holds.

Proof. (i) Let us say that t is presteep if it satisfies the condition of (i). By definition, if a appears in t , then $\bar{t} \sqsupset a$ holds, and, therefore, t being steep implies t being presteep. On the other hand, if t' is an LD-expansion of t , and t' is presteep, then t is presteep as well, since every element appearing in t appears in t' . So, in order to prove that presteepness coincides with steepness, it suffices to prove that every LD-expansion of a presteep term is presteep, and, for an induction, it suffices to assume that t is presteep and to consider the case $t' = (t)\alpha$. By Lemma 1.8, the elements that appear in t' are those elements that appear in t , completed with new elements of the form ab , where a appears at $\alpha 101^*$ in t , and b appears below $\alpha 0$ in t . As $\bar{t}' = \bar{t}$ holds, it suffices to verify the steepness condition for the latter elements. Now, we have in this case

$$\bar{t} \gg a\bar{t} \gg ab^{[2]} = (ab)^{[2]}.$$

The first inequality is a consequence of $\bar{t} \gg a^{[2]}$, as $\bar{t} \sqsupset a$ implies $a\bar{t} \sqsupset a^{[2]}$, and, therefore, by Proposition 1.5(ii), $\bar{t} \gg a^{[2]}$ implies $\bar{t} \gg a\bar{t}$.

(ii) Assume that t satisfies the conditions and a appears in t . Three cases are possible. If a appears in t_1 , then the assumption that t_1 is steep gives $\bar{t}_1 \gg a^{[2]}$, and we deduce $\bar{t}_1 t_2 \gg a^{[2]}$. If $a = \bar{t}_1$ holds, then the hypothesis $\bar{t}_2 \gg \bar{t}_1$ implies $\bar{t}_1 t_2 \gg \bar{t}_1 \bar{t}_1 = a^{[2]}$. Finally, if $a = \bar{t}_1 b$ holds, with b appearing in t_2 , the assumption that t_2 is steep gives $\bar{t}_2 \gg b^{[2]}$, and we deduce $\bar{t}_1 t_2 \gg \bar{t}_1 b^{[2]} = a^{[2]}$. By (i), this is enough to conclude that t is steep. ■

Lemma 2.3. Every quasi-increasing term is steep.

Proof. We use induction on t . If t is a variable, the result is vacuously true. Assume $t = t_1 t_2$. By definition, every subterm of a quasi-increasing term is quasi-increasing. Hence, by induction hypothesis, t_1 and t_2 are steep. Moreover, the quasi-increasing condition at ϕ in t gives $\text{var}(t, 10^*) > \text{var}(t, 0^*)$. Hence $t_2 \gg t_1$ holds, and Proposition 2.2(ii) applies. ■

Steep elements enjoy the following order constraints that will be used several times in the sequel.

Lemma 2.4. Assume that a is steep and the inequalities $a > a_1, \dots, a > a_p$ hold. Then $a > a_1 \cdots a_p$ holds, and $a \sqsupset a_1 \cdots a_p$ is possible only if $a \sqsupset a_i \sqsupset a_{i+1} \cdots a_p$ holds for every i .

Proof. We use induction on $p \geq 1$. For $p = 1$, the result is obvious. Assume $p \geq 2$. If $a \gg a_1$ holds, then $a \gg a_1 a_2 \cdots a_p$ does. Assume now $a \sqsupset a_1$. The assumption that a is steep gives $a \gg a_1^{[2]}$. Now, $a \sqsupset a_1$ implies $a_1 a \sqsupset a_1^{[2]}$, and, therefore, $a \gg a_1 a$. By induction hypothesis, we have $a > a_2 \cdots a_p$, so we deduce $a \gg a_1 a > a_1 a_2 \cdots a_p$, which implies $a > a_1 a_2 \cdots a_p$. Moreover, if $a \gg a_2 \cdots a_p$ holds, we find $a \gg a_1 a \gg a_1 a_2 \cdots a_p$, hence $a \gg a_1 a_2 \cdots a_p$.

Assume now $a \sqsupset a_1 a_2 \cdots a_p$. The previous argument shows that the only possible case is $a \sqsupset a_1$ and $a \sqsupset a_2 \cdots a_p$. As a_1 and $a_2 \cdots a_p$ both are \sqsubseteq -smaller than a , they must be \sqsubseteq -comparable. Now, the inequality $a_2 \cdots a_p \sqsupseteq a_1$ is impossible, as it would imply $a \sqsupset a_1 a_2 \cdots a_p \sqsupseteq a_1^{[2]}$, contradicting the steepness of a . Hence, we have $a_1 \sqsupset a_2 \cdots a_p$, and, therefore, $a \sqsupset a_2 \cdots a_p$. By induction hypothesis, we deduce $a_i \sqsupset a_{i+1} \cdots a_p$ for $2 \leq i < p$. ■

Products of cuts

The following question will be crucial: if t is a term, and a, b appear in t , does ab appear in t , or, more generally, in some LD-expansion of t ? The problem is very difficult in general, but we shall establish a complete answer for a family of particular terms called perfect terms. We start from the following observation:

Proposition 2.5. *Assume that t is a term in T_∞ , and a is an element of FLD_∞ . The following are equivalent:*

- (i) *The inequality $\bar{t} \sqsupset a$ holds;*
- (ii) *The element a appears in some term t' that is LD-equivalent to t .*
- (iii) *The element a appears in some term $\partial^k t$.*

Proof. By construction, ∂t is LD-equivalent to t , so (iii) implies (ii), and, as $t' =_{LD} t$ means $\bar{t}' = \bar{t}$, (ii) implies (i). Hence the point is to prove that (i) implies (iii). So, assume that s is a term, and $t \sqsupset_{LD} s$ holds. By definition, there exists a term t' and an integer p' satisfying $t' =_{LD} t$ and $s =_{LD} \text{sub}(t', 0^{p'})$. By the confluence property, there exists k such that $\partial^k t$ is an LD-expansion of t' . Then, by Proposition 1.3, there exists $p \geq p'$ satisfying $s =_{LD} \text{sub}(\partial^k t, 0^p)$, which means that the class a of s appears in $\partial^k t$ at 0^p . ■

Assume that t is a term and a appears in t . A first, trivial observation is that, for a given a , the set of those b 's such that ab appears in an LD-expansion of t make an initial segment with respect to the \sqsubset -ordering. Indeed, ab appears in some LD-expansion of t if and only if $ab \sqsubset \bar{t}$ holds, and the latter relation implies $ab' \sqsubset \bar{t}$ whenever $b \sqsupseteq b'$ holds.

Now, assuming that a and b appear in t , we shall consider three cases according to the position of b in the order of FLD_∞ , and prove the following results:

- If b is very small, then ab always appears in some LD-expansion of t , actually in ∂t ;
- If b is large, and t is steep, then ab never appears in any LD-expansion of t ;
- In the intermediate case, and if t is what we shall call full, then ab appears in some LD-expansion of t if and only if it appears in t , and there exists an effective algorithm to determine if this is the case.

The first two cases are easy.

Lemma 2.6. *Assume that t is a term, and a uncovers b in t . Then ab does not appear in t , but it appears in ∂t .*

Proof. Assume that a appears at α and b appears at β in t . By hypothesis, (α, β) is a descent in t . Let γ be the address $\pi_t((\alpha, \beta))$. By (1.3), we have

$$\text{cut}(\partial t, \gamma) =_{LD} \text{cut}(t, \alpha)\text{cut}(t, \beta),$$

and ab appears at γ in ∂t . The injectivity of the mapping π_t implies that γ cannot be written as $\pi_t((\gamma_0))$ for any address γ_0 in t . This shows that the LD-class of $\text{cut}(\partial t, \gamma)$ does not appear in t . ■

Lemma 2.7. *Assume that t is a steep term, a, b appear in t , and $a \leq b$ holds. Then ab appears in no LD-expansion of t .*

Proof. As a and b appear in t , $a \leq b$ implies $b \sqsupseteq a$, and, therefore, $ab \sqsupseteq a^{[2]}$. As t is steep, we have $\bar{t} \gg a^{[2]}$, hence, by Proposition 2.4(ii), $\bar{t} \gg ab$, which forbids $\bar{t} \sqsupset ab$. ■

Observe that the previous argument works with the weaker hypothesis that a and b appear in some LD-expansion of t only.

Full terms

We turn to the difficult case, namely when a covers b in t . Examples show that ab may not appear in t , but appear in some LD-expansion $\partial^k t$ with $k \geq 1$, and we have no control of the situation in the most general case. So we shall concentrate on particular terms. A natural hypothesis for discarding the problems is to require that ab , if it appears in some LD-expansion of t , already appears in t .

Definition. Assume that t is a term. We say that t is *full* if ab appears in t whenever a covers b in t and $\bar{t} \sqsupset ab$ holds.

Example 2.8. Let $x^{[m]}$ be the term inductively defined by $x^{[1]} = x$, $x^{[m]} = xx^{[m-1]}$ for $m \geq 2$. For every m , $x^{[m]}$ is full. Indeed, no non-final address in the outline of $x^{[m]}$ covers any address, so the fullness condition is vacuously true. Let us consider similarly the term $\partial x^{[m]}$. The skeleton of $\partial x^{[m]}$ is a complete binary tree of height m . By Formula (1.3), those elements that appear in $\partial x^{[m]}$ have the form $x^{[i_1]} \dots x^{[i_{p+1}]}$ with $i_1 > \dots > i_{p+1}$. Let a, b two elements appearing in $\partial x^{[m]}$, say $a = x^{[i_1]} \dots x^{[i_{p+1}]}$, $b = x^{[j_1]} \dots x^{[j_{q+1}]}$. By Proposition 1.10, a covers b in $\partial x^{[m]}$ if and only if we have $p \leq q$ and $i_1 = j_1, \dots, i_p = j_p, i_{p+1} > j_{p+1}$. Then we have $ab = x^{[i_1]} \dots x^{[i_p]} x^{[j_{p+1}]} \dots x^{[j_{q+1}]}$, which appears in $\partial x^{[m]}$. Hence every term $\partial x^{[m]}$ is full. On the other hand, it follows from the results of [11] that the term $\partial^2 x^{[6]}$ is not full.

Lemma 2.9. *Every cut of a full term is full.*

Proof. Assume that t is full, and t' is a cut of t . Assume that a covers b in t' and $\bar{t}' \sqsupset ab$ holds. By definition of covering, a covers b in t , and $\bar{t} \sqsupset ab$ holds by transitivity of \sqsupset . As t is full, ab appears in t . Now $\bar{t}' \sqsupset ab$ implies that ab appears in t on the left of \bar{t}' , hence it appears in t' as well. ■

Let us say that the term t_1 is a substitute of the term t if t_1 is the image of t under some endomorphism f of T_∞ into itself; in this case, we write $t_1 = t^f$. Every such substitution f of T_∞ induces a well defined endomorphism of FLD_∞ ; we use a^f for the image of a under this endomorphism.

Lemma 2.10. *If a covers b in t , then, for every substitution f , a^f covers b^f in t^f .*

Proof. Assume that a appears in t at α . Then a^f appears in t^f at $\alpha 1^p$, where p is the right height of the term $\text{var}(t, \alpha)^f$. Similarly, if b appears at β in t , then b^f appears in t^f at some address of the form $\beta 1^q$. By definition, α covering β implies that $\alpha 1^p$ covers $\beta 1^q$, and, therefore, a^f covers b^f in t^f . ■

Proposition 2.11. *Assume that some substitute of t is full. Then t is full as well.*

Proof. Assume that f is a substitution, and the term t^f is full. Assume that a covers b in t , and $\bar{t} \sqsupset ab$ holds. By Proposition 2.5, ab appears in some term $\partial^k t$ at some address say α . Let $x_i = \text{var}(\partial^k t, \alpha)$. Then, by construction, $a^f b^f$ occurs in $(\partial^k t)^f$ at $\alpha 1^p$, where 1^p belongs to the outline of $f(x_i)$.

On the other hand, by Lemma 2.10, a^f covers b^f in t^f . Moreover, the hypothesis $\bar{t} \sqsupset ab$ implies $\bar{t}^f \sqsupset a^f b^f$, as f is an endomorphism and \sqsupset is definable from the product and,

therefore, preserved under f . As the term t^f is full, we deduce that $a^f b^f$ appears in t^f , at some address, say $\beta\gamma$, where β belongs to the outline of t . As $\partial^k t$ is an LD-expansion of t , $a^f b^f$ appears in $(\partial^k t)^f$ at the address $\beta'\gamma$, where β' is the heir of β in $\partial^k t$. Let x_j be the variable that occurs at β in t . The comparison of the above results implies $\alpha 1^p = \beta'\gamma$, hence $j = i$, $\beta' = \alpha$ and $\gamma = 1^p$. We deduce $\text{cut}(\partial^k t, \alpha)^f =_{LD} \text{cut}(t, \beta)^f$, hence $\text{cut}(\partial^k t, \alpha) =_{LD} \text{cut}(t, \beta)$: otherwise, we would have $\text{cut}(\partial^k t, \alpha) \sqsubset_{LD} \text{cut}(t, \beta)$ or $\text{cut}(\partial^k t, \alpha) \sqsupset_{LD} \text{cut}(t, \beta)$ since the cuts of $\partial^k t$ are pairwise \sqsubset_{LD} -comparable, and this would imply $\text{cut}(\partial^k t, \alpha)^f \sqsubset_{LD} \text{cut}(t, \beta)^f$ or $\text{cut}(\partial^k t, \alpha)^f \sqsupset_{LD} \text{cut}(t, \beta)^f$. Now, this means that some cut of t is LD-equivalent to $\text{cut}(\partial^k a, \alpha)$, *i.e.*, that ab appears in t . Hence t is full. ■

Perfect terms

We are interested in proving fullness results for large families of terms. As was mentioned in Example 2.8, the term ∂t need not be full even if t is. A good hypothesis is to require steepness and fullness simultaneously. In some sense, these hypotheses are complementary: when $\bar{t} \sqsupset abc$ is assumed, steepness implies $a \sqsupset bc$, *i.e.*, it says that bc is not too large, while, by definition, fullness implies (when a covers bc) that abc appears in t , so it says that bc is not too small.

Definition. Assume that t is a term. We say that t is *perfect* if it is both steep and full.

Proposition 2.12. *Every quasi-increasing term is perfect.*

Proof. Assume that t is quasi-increasing. By Lemma 2.3, t is steep. Assume that a covers b in t . By Lemma 2.1, $\bar{t} \sqsupset ab$ does not hold, and the fullness condition is vacuously true in t . ■

The following technical result is crucial.

Lemma 2.13. *Assume that t is a perfect term, and we have $\bar{t} \sqsupset a$, $\bar{t} \sqsupset b$, $\bar{t} \sqsupset c$, and $\bar{t} \sqsupset abc$. Assume moreover that ab and bc appear in t . Then ac and abc appear in t as well.*

Proof. The term t is steep, hence, by Lemma 2.4, $\bar{t} \sqsupset abc$ implies $a \sqsupset bc$. As ab appears in t , a covers b in t (by Lemma 2.6), and, therefore, it covers bc as well, since the latter appears on the right of b in t . As t is full, we deduce that $a(bc)$ appears in t .

As for ac , we know that $b \sqsupset c$ holds, since bc appears in t , hence we have $\bar{t} \sqsupset ab \sqsupset ac$. Moreover, a covers b in t since ab appears in t , and, similarly, b covers c in t . By transitivity of covering, a covers c in t . As t is full, this implies that ac appears in t . ■

We are going to prove now that perfectness is preserved under operation ∂ . To this end, we must be able to determine those elements that appear in ∂t in terms of those elements that appear in t .

Definition. Assume that t is a term and a appears in ∂t . By Formula (1.3), there exists a decreasing sequence (a_1, \dots, a_{p+1}) such that $a = a_1 \cdots a_{p+1}$ holds and a_i uncovers a_{i+1} in t for every i , and this sequence is unique as the mapping π_t is bijective: it will be called the *t -decomposition of a* .

Lemma 2.14. *Assume that a covers b in ∂t , and (a_1, \dots, a_{p+1}) is the t -decomposition of a . Then the t -decomposition of b has the form $(a_1, \dots, a_p, b_{p+1}, \dots, b_{q+1})$ with $a_{p+1} > b_{p+1}$.*

Proof. Assume that a appears at α in t , and that b appears at β . Let $(\alpha_1, \dots, \alpha_{p+1})$ and $(\beta_1, \dots, \beta_{q+1})$ be the preimages of α and β under the mapping π_t . The last statement in Proposition 1.10 says that α covers β if and only if $p \leq q$ holds, the first p terms of $(\beta_1, \dots, \beta_{q+1})$ coincide with the first p terms of $(\alpha_1, \dots, \alpha_{p+1})$, and $\alpha_{p+1} >_{LR} \beta_{p+1}$ holds. Introducing a_i to be the class of $\text{cut}(t, \alpha_i)$ and b_j to be the class of $\text{cut}(t, \beta_j)$ gives the result. \blacksquare

The problem is as follows: We assume that t is a perfect term, and we try to prove that ∂t is perfect as well. So, we assume that a covers b in ∂t , and study whether ab appears in ∂t . By the previous lemma, we know that the t -decompositions of a and b have the form (a_1, \dots, a_{p+1}) and $(a_1, \dots, a_p, b_{p+1}, \dots, b_{q+1})$ with $a_{p+1} > b_{p+1}$. Using left self-distributivity, we find

$$ab = (a_1 \cdots a_p a_{p+1})(a_1 \cdots a_p b_{p+1} \cdots b_{q+1}) = a_1 \cdots a_p a_{p+1} b_{p+1} \cdots b_{q+1}. \quad (2.1)$$

The sequence involved in the latter expression is almost the t -decomposition of some element appearing in t : if a_{p+1} uncovers b_{p+1} in t , *i.e.*, if the sequence $(\alpha_1, \dots, \alpha_p, \alpha_{p+1}, \beta_{p+1}, \dots, \beta_{q+1})$ is a descent in t , then, letting γ be the image of this descent under π_t , we see that ab appears at γ in ∂t . Now, if a_{p+1} covers b_{p+1} in t , (2.1) is no longer the t -decomposition of an element appearing in t . But, in this case, the hypothesis that t is perfect implies that the element $a_{p+1}b_{p+1}$ appears in t , and we can replace the sequence of (2.1) with a new sequence. This leads us to the notion of t -reduction of a sequence.

Definition. Assume that \vec{a}, \vec{a}' are finite nonempty sequences of elements of FLD_∞ appearing in the term t . We say that \vec{a} is t -reducible to \vec{a}' in one step if \vec{a} and \vec{a}' either have the form

$$\begin{cases} \vec{a} = (a_1, \dots, a_{i-1}, a_i, a_{i+1}, a_{i+2}, \dots, a_{r+1}), \\ \vec{a}' = (a_1, \dots, a_{i-1}, a_i a_{i+1}, a_i, a_{i+2}, \dots, a_{r+1}), \end{cases}$$

for some $i < r$, or they have the form

$$\begin{cases} \vec{a} = (a_1, \dots, a_{i-1}, a_r, a_{r+1}), \\ \vec{a}' = (a_1, \dots, a_{i-1}, a_r a_{r+1}). \end{cases}$$

We say that \vec{a} is t -reducible to \vec{a}' in n steps if there exists a length $n + 1$ sequence from \vec{a} to \vec{a}' such that every component is t -reducible to the next one in one step, and that \vec{a} is t -irreducible if it is t -reducible to no sequence but itself.

The idea of reduction is to use the product of FLD_∞ to eliminate the covering patterns, and to iterate the process until no more covering remains, if this is possible. Several verifications are needed. For \vec{a} a sequence as above, say $\vec{a} = (a_1, \dots, a_r)$, we write $\prod \vec{a}$ for $a_1 \cdots a_r$.

Lemma 2.15. Assume that t is a term with n occurrences of variables.

- (i) Assume that \vec{a} is t -reducible to \vec{a}' . Then $\prod \vec{a}' = \prod \vec{a}$ holds.
- (ii) Every sequence of t -reductions starting with a length r sequence converges to a t -irreducible sequence in $(n - 1)^r$ steps at most.

Proof. Point (i) follows from the definition of reduction. For (ii), we observe that, if \vec{a} is t -reducible to \vec{a}' , then the sequence \vec{a}' is larger than \vec{a} in the lexicographic extension of \sqsubset to sequences from FLD_∞ , and the length of \vec{a}' is at most the length of \vec{a} . By definition, $n - 1$ elements appear in the term t . So, $(n - 1)^r$ different sequences at most appear in a sequence of t -reductions starting with a length r sequence \vec{a} , and, therefore, any sequence of t -reductions from \vec{a} results in a t -irreducible sequence after $(n - 1)^r$ steps at most. \blacksquare

With our definition, t -reduction is not a functional process: a given sequence may be t -reducible to several distinct sequences. However, in good cases, the choice of the t -reduction steps does not matter.

Lemma 2.16. *Assume that t is a perfect term, and \vec{a} is a sequence of elements appearing in t and satisfying $\vec{t} \sqsupset \prod \vec{a}$. Then t -reduction from \vec{a} is confluent, i.e., if the sequence \vec{a} is t -reducible both to \vec{a}_1 and \vec{a}_2 , there exists a sequence \vec{a}' such that both \vec{a}_1 and \vec{a}_2 are t -reducible to \vec{a}' .*

Proof. As t -reduction is Noetherian, i.e., it has no infinite descending sequence, it suffices to prove that it is locally confluent, i.e., to show that, if \vec{a} is t -reducible to \vec{a}_0 , and \vec{a}_0 is t -reducible in one step to \vec{a}_1 and \vec{a}_2 , then there exists a sequence \vec{a}' such that both \vec{a}_1 and \vec{a}_2 are t -reducible to \vec{a}' . Assume $\vec{a}_0 = (a_1, \dots, a_{r+1})$, and \vec{a}_1 and \vec{a}_2 are obtained from \vec{a} using reduction at positions i and j respectively. For $|i - j| \geq 2$, the reductions involve disjoint fragments of the sequences, and we can define \vec{a}' to be the sequence obtained from \vec{a}_0 by reducing both at i and at j . The critical case is $|i - j| = 1$. Assume for instance $i + 1 = j < r$. Then we have

$$\begin{aligned}\vec{a}_1 &= (\dots, a_{i-1}, a_i a_{i+1}, a_i, a_{i+2}, a_{i+3}, \dots), \\ \vec{a}_2 &= (\dots, a_{i-1}, a_i, a_{i+1} a_{i+2}, a_{i+1}, a_{i+3}, \dots),\end{aligned}$$

By Lemma 2.15, we have $\prod \vec{a}_0 = \prod \vec{a}$, and, therefore, $\vec{t} \sqsupset \prod \vec{a}_0$. By Lemma 2.4, we deduce $\vec{t} \sqsupset a_i a_{i+1} a_{i+2} \dots$, and, therefore, $\vec{t} \sqsupset a_i a_{i+1} a_{i+2}$. Applying Lemma 2.13, we see that $a_i a_{i+2}$ and $a_i a_{i+1} a_{i+2}$ appear in t as well. This implies that the sequences \vec{a}_1 and \vec{a}_2 both are t -reducible to the sequence

$$\vec{a}' = (\dots, a_{i-1}, a_i a_{i+1} a_{i+2}, a_i a_{i+1}, a_i, a_{i+3}, \dots).$$

The case $i + 1 = j = r$ is similar, with the terminal sequence $(\dots, a_{i-1}, a_i a_{i+1} a_{i+2})$. ■

Proposition 2.17. *Assume that t is a perfect term, and \vec{a} is a sequence of elements appearing in t and satisfying $\vec{t} \sqsupset \prod \vec{a}$. Then t -reduction from \vec{a} leads to a unique t -irreducible sequence, which is the t -decomposition of an element of ∂t . In particular, $\prod \vec{a}$ appears in ∂t .*

Proof. Consider an arbitrary sequence of t -reductions from \vec{a} . By Lemma 2.15, it must have a finite length and end with some t -irreducible sequence say \vec{a}' . Assume that \vec{a} is t -reducible to \vec{b} . By Lemma 2.16, \vec{a}' and \vec{b} have to be t -reducible to some sequence \vec{c} , and the hypothesis that \vec{a}' is t -irreducible implies that \vec{c} coincides with \vec{a}' , i.e., \vec{b} is t -reducible to \vec{a}' . Hence \vec{a}' is the unique t -irreducible sequence accessible from \vec{a} . Assume $\vec{a}' = (a'_1, \dots, a'_{r+1})$. By Lemma 2.4, $\vec{t} \sqsupset \prod \vec{a}'$ implies \vec{a}' being strictly decreasing. Assume that a'_i covers a'_{i+1} for some i . Then, $\vec{t} \sqsupset a'_1 \dots a'_i a'_{i+1}$ and t being perfect imply that $a'_i a'_{i+1}$ appears in t , which contradicts the hypothesis of \vec{a}' being t -irreducible. Hence we conclude that a'_i uncovers a'_{i+1} in t for every i : this means that \vec{a}' is the t -decomposition of an element of ∂t . ■

In the above framework, we denote by $\text{red}_t(\vec{a})$ the unique t -irreducible sequence obtained from \vec{a} using t -reduction. We can describe the elements ab that appear in ∂t completely.

Lemma 2.18. *Assume that t is a perfect term, that a covers b in ∂t , and that $\vec{t} \sqsupset ab$ holds. Then the element ab appears in ∂t , and, letting (a_1, \dots, a_{p+1}) and (b_1, \dots, b_{q+1}) be respectively the t -decompositions of a and b , the t -decomposition of ab is $\text{red}_t(a_1, \dots, a_{p+1}, b_{p+1}, \dots, b_{q+1})$.*

Proof. By Lemma 2.14, the t -decompositions of a and b have the form $(a_1, \dots, a_p, a_{p+1})$ and $(a_1, \dots, a_p, b_{p+1}, \dots, b_{q+1})$ respectively. Let $\vec{c} = (a_1, \dots, a_{p+1}, b_{p+1}, \dots, b_{q+1})$. By construction, the elements of \vec{c} appear in t , and we have $\tilde{t} \sqsupset \prod \vec{c} = ab$. By Proposition 2.17, the sequence $\text{red}_t(\vec{c})$ is the t -decomposition of an element of ∂t . This means that ab appears in ∂t , and $\text{red}_t(\vec{c})$ is its t -decomposition. ■

Assume that the term t is perfect. Then the term ∂t is steep, as it is LD-equivalent to t , and Lemma 2.18 tells us that ∂t is full. Thus, we have completed a proof of:

Proposition 2.19. *If t is a perfect term, then $\partial^k t$ is perfect for every k .*

Definition. The term t is said to be *quasi-injective* if there exists a quasi-increasing term t_0 in T_∞ and a bijection f of the variables of t_0 onto the variables of t satisfying $t = t_0^f$.

Thus a quasi-injective term is a quasi-increasing term up to a permutation of the variables. In particular, every injective term is quasi-injective. We can state:

Proposition 2.20. *If t is a quasi-injective term, then $\partial^k t$ is full for every k .*

Proof. Assume $t = t_0^f$, where t_0 is quasi-increasing and f is a permutation of variables. By Lemma 2.3, t_0 is steep, and, by Proposition 2.12, it is perfect. By Proposition 2.19, every term $\partial^k t_0$ is perfect, hence full. Finally, fullness involves Identity (*LD*) only, and, therefore, it is preserved under renaming the variables: thus $\partial^k t$ is full as well. ■

3. CONVERGENCE RESULTS FOR THE POLISH ALGORITHM

The Polish algorithm is a syntactic method for deciding LD-equivalence of terms. It has been investigated in [3] and [11], and it was considered independently by several researchers. The problem of whether it always converges is open. The aim of this section is to prove partial convergence results, which go (far) beyond all previously known results.

The Polish Algorithm

In this introductory subsection, we recall the definition of the Polish Algorithm [10]. As our notation here is different, we shall reprove the few simple results about LD-expansions the construction relies upon.

We recall that, for t a term, \tilde{t} denotes the right Polish notation of t .

Definition. Assume that t_1, t_2 are terms in T_∞ . We say that t_1 and t_2 admit a *hard clash* at position p if either there exist a length $p-1$ word w such that \tilde{t}_1 begins with wx_i and \tilde{t}_2 begins with wx_j with $i \neq j$, or \tilde{t}_1 is a proper prefix of \tilde{t}_2 and it has length $p-1$, or *vice versa*; We say that t_1 and t_2 admit a *soft clash* at position p if there exist a length $p-1$ word w such that \tilde{t}_1 begins with wx_i and \tilde{t}_2 begins with $w\bullet$, or *vice versa*.

For instance, the terms $(x_1x_2)(x_1x_3x_4)$ and $x_1((x_2x_3)(x_2x_4))$ admit a soft clash at position 3, since their right Polish forms are $x_1x_2\bullet x_1x_3x_4\bullet\bullet\bullet$ and $x_1x_2x_3\bullet x_2x_4\bullet\bullet\bullet$ respectively.

By definition, the terms t_1 and t_2 have a hard clash if and only if one of $t_1 \ll t_2$, $t_2 \ll t_1$, $t_1 \sqsubset t_2$, $t_2 \sqsubset t_1$ holds, and, in all cases, they are LD-inequivalent by Proposition 2.4. Hence the only case when we cannot directly decide whether t_1 and t_2 are LD-equivalent is the case of a soft clash. Now, in this case, there happens to exist a canonical method for eliminating the clash.

It follows from the construction of the right Polish form that, for every term t , there exists a one-to-one correspondence between the letters in the word \tilde{t} and the nodes in the tree associated with t , the variables corresponding to the leaves of the tree, and the letters \bullet corresponding to the internal nodes. The inductive definition is straightforward. For each address α in the skeleton of t , the associated position in the word \tilde{t} will be simply called *the* position of α in \tilde{t} . For instance, by definition of the right Polish form, the position of the address ϕ (the root of the tree) is always the last position in \tilde{t} . A significant point is that the correspondence is compatible with the orderings in the sense that, if the address α lies on the right of the address β in the outline of the term t , then the position of α is larger than the position of β . We recall that, for α in the skeleton of t , we use $\alpha 0^*$ for the unique addresses of the form $\alpha 0^p$ lying in the outline of t ; similarly, we use $\alpha 1^*$ for the address $\alpha 1^q$ lying in the outline of t (this convention avoids introducing the exponents p, q explicitly).

Lemma 3.1. *Assume that t' is a proper LD-expansion of t . Then t and t' have a soft clash. More precisely, assume that t' is the basic LD-expansion $(t)\alpha$. Then t and t' have a soft clash at p , where p is the position of $\alpha 110^*$ in t and the position of $\alpha 0^*$ in t' .*

Proof. Assume $t' = (t)\alpha$. Let s_1, s_2 and s_3 be the subterms of t at $\alpha 0$, $\alpha 10$, and $\alpha 11$ respectively. Then we have explicit decompositions

$$\tilde{t} = \dots \tilde{s}_1 \tilde{s}_2 \tilde{s}_3 \bullet \bullet \dots, \quad \tilde{t}' = \dots \tilde{s}_1 \tilde{s}_2 \bullet \tilde{s}_1 \tilde{s}_3 \bullet \bullet \dots,$$

so t and t' have a soft clash at the position which corresponds to the first letter of \tilde{s}_3 in \tilde{t} , *i.e.*, to the position of $\alpha 110^*$ in t , and to the first letter \bullet after \tilde{s}_2 in \tilde{t}' , *i.e.*, to the position of $\alpha 0^*$ in t' . ■

Assume that t_1 and t_2 have a soft clash at position p , say for instance that \bullet occurs at position p in t_1 . Expanding t_2 using (LD) gives a new term t'_2 that has a soft clash with t_2 . The point is that there always exists a way to choose the LD-expansion so that the clash between t_2 and t'_2 lies at position p and the possible clash between t_1 and t'_2 lies at position $p + 1$ at least.

Example 3.2. Let t_1 and t_2 be the terms considered above, namely

$$\begin{cases} \tilde{t}_1 = x_1x_2\bullet x_1x_3x_4\bullet\bullet\bullet, \\ \tilde{t}_2 = x_1x_2x_3\bullet x_2x_4\bullet\bullet\bullet. \end{cases}$$

They admit a soft clash at position 3. Let t'_2 be the LD-expansion $((t_2)\phi)0$. We have

$$\begin{cases} \tilde{t}_1 = x_1x_2\bullet x_1x_3x_4\bullet\bullet\bullet, \\ \tilde{t}'_2 = x_1x_2\bullet x_1x_3\bullet\bullet x_1x_2x_4\bullet\bullet\bullet, \end{cases}$$

and t_1 and t'_2 have a soft clash at position 6.

In order to describe the construction precisely, we need some notation. For each address α , we have introduced, for t a term that is large enough, $(t)\alpha$ to be the LD-expansion of t corresponding to applying left self-distributivity at α in t . For $\alpha_1 \dots \alpha_p$ a finite sequence of addresses, *i.e.*, a word on the set \mathbf{A} of all addresses, we denote similarly by $(t)\alpha_1 \dots \alpha_p$ the iterated LD-expansion $(\dots ((t)\alpha_1)\alpha_2 \dots)\alpha_p$. We thus obtain a partial action on the right of the free monoid \mathbf{A}^* generated by \mathbf{A} on the set T_∞ : this action is partial, as $(t)w$ is defined only if the skeleton of t is large enough.

Definition. For α an address, and $k \geq 0$, the word $\alpha_{(k)}$ is defined to be the empty word ε for $k = 0$, and to be $\alpha \cdot \alpha 0 \cdot \dots \cdot \alpha 0^{k-1}$ otherwise.

For instance, the word $\phi \cdot 0$ involved in Example 3.2 above is $\phi_{(2)}$.

Lemma 3.3. Assume that t is a term. Then $(t)\alpha_{(k)}$ is defined if and only if the address $\alpha 10^k$ belongs to the skeleton of t . In this case, the terms t and $(t)\alpha_{(k)}$ have a soft clash at the position of $\alpha 10^{k-1} 10^*$ in t , which is also the position of $\alpha 0^k$ in $(t)\alpha_{(k)}$.

Proof. Assume $t' = (t)\alpha_{(k)}$. An induction on k shows that the $\alpha 10^k$ -subterm of t is defined, and that, letting s_1 denote the $\alpha 0$ -subterm of t , s_2 denote its $\alpha 10^k$ -subterm, and, for $2 \leq j \leq k+1$, s_j denote its $\alpha 10^{k-j+2} 1$ -subterm, we have

$$\begin{cases} \text{sub}(\widetilde{t}, \alpha) = \widetilde{s}_1 \widetilde{s}_2 \widetilde{s}_3 \bullet \dots \widetilde{s}_{k+1} \bullet \widetilde{s}_{k+2} \bullet \bullet, \\ \text{sub}(t', \alpha) = \widetilde{s}_1 \widetilde{s}_2 \bullet \widetilde{s}_1 \widetilde{s}_3 \bullet \bullet \dots \widetilde{s}_1 \widetilde{s}_{k+1} \bullet \bullet \widetilde{s}_1 \widetilde{s}_{k+2} \bullet \bullet, \end{cases}$$

i.e., t' is obtained from t by distributing s_1 to s_3, \dots, s_{k+2} (Figure 3.1). Let w be the prefix of the word \widetilde{t} that ends with $\widetilde{s}_1 \widetilde{s}_2$, and let $p-1$ be the length of w . Then, in \widetilde{t} , w is followed by the leftmost variable x_i of s_3 , while, in t' , it is followed by the letter \bullet whose address is $\alpha 0^k$. Hence, by definition, t and t' have a soft clash at position p . The addresses of p in t and t' can be read on Figure 3.1. ■

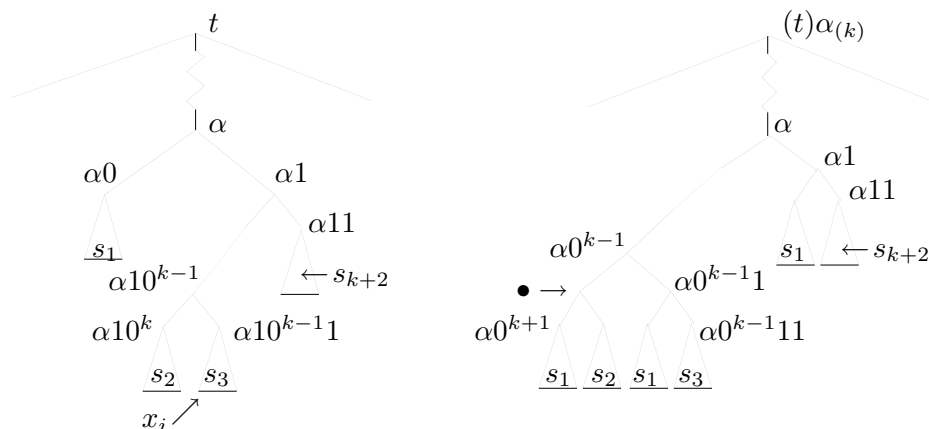


Figure 3.1. Comparing t and $(t)\alpha_{(k)}$

Lemma 3.4. Assume that the terms t_1 and t_2 have a soft clash at p , with \bullet at position p in \widetilde{t}_1 . Then the address of p in t_2 has the form $\alpha 10^{k-1} 10^*$, the length p prefixes of the words \widetilde{t}_1 and $(t_2)\alpha_{(k)}$ coincide, and $\alpha_{(k)}$ is the only word of the form $\beta_{(j)}$ with this property.

Proof. By hypothesis, the words \widetilde{t}_1 and \widetilde{t}_2 have the same length $p-1$ prefix w , and the p -th letter in \widetilde{t}_1 is \bullet , while the p -th letter in \widetilde{t}_2 is some variable x_i . Let γ_e be the address of p in t_e , for $e = 1, 2$. Proving that γ_2 has the form $\alpha 10^{k-1} 10^*$ means proving that it contains at least two 1's. By construction, the number of 1's in γ_2 is equal to $\#_x(wx_i) - \#_\bullet(wx_i) - 1$, *i.e.*, to $\#_x(w) - \#_\bullet(w)$, where $\#_x(u)$ and $\#_\bullet(u)$ respectively denote the number of variables and of \bullet 's in u . For the same reason, the number of 1's in γ_1 is $\#_x(w\bullet) - \#_\bullet(w\bullet) - 1$, which is $\#_x(w) - \#_\bullet(w) - 2$. By construction of the right Polish form, the latter number must be nonnegative, and we deduce $\#_x(wx_i) - \#_\bullet(wx_i) - 1 \geq 2$.

Then Lemma 3.3 shows that $(t_2)\alpha_{(k)}$ is defined, that t_2 and $(t_2)\alpha_{(k)}$ have a soft clash at p , and that, for $\beta \neq \alpha$ or $j \neq k$, the clash between t_2 and $(t_2)\beta_{(j)}$ is not at p . ■

The natural idea is now to use Lemma 3.4 to push the soft clash to the right, and to iterate the process until either equal terms are obtained, or until a hard clash appears.

Definition. [10] Assume that the terms t_1, t_2 have a soft clash at p , and \bullet occurs at position p in t_1 . We say that the Polish Algorithm running on (t_1, t_2) returns (t'_1, t'_2) in one step, with output $(\varepsilon, \alpha_{(k)})$, if we have $t'_1 = t_1$, and $t'_2 = (t_2)\alpha_{(k)}$, where $\alpha 10^{k-1} 10^*$ is the address of p in t_2 . The definition when \bullet occurs at p in \tilde{t}_2 is symmetric.

For $m \geq 0$, we say that the Polish Algorithm running on (t_1, t_2) returns (t'_1, t'_2) in m steps, with output (u_1, u_2) if there exist two sequences of terms $t_e^{(0)} = t_1, t_e^{(1)}, \dots, t_e^{(m)} = t'_e, e = 1, 2$, such that, for every i , the Polish Algorithm running on $(t_1^{(i)}, t_2^{(i)})$ returns $(t_1^{(i+1)}, t_2^{(i+1)})$ in one step, and (u_1, u_2) is obtained by concatenating the outputs at each step.

Finally, we say that the Polish Algorithm running on (t_1, t_2) converges to (t'_1, t'_2) in m steps if it returns (t'_1, t'_2) in m steps and t'_1 and t'_2 do not have a soft clash.

The following fact results from the definition directly:

Lemma 3.5. Assume that the Polish Algorithm running on the pair (t_1, t_2) returns the pair (t'_1, t'_2) with output (u_1, u_2) . Then we have $t'_1 = (t_1)u_1$ and $t'_2 = (t_2)u_2$.

The reader can check that the Polish Algorithm running on the pair (t_1, t_2) of Example 3.2 converges in 4 steps to the pair (t', t') , with $t' = ((x_1x_2)(x_1x_3))((x_1x_2)(x_1x_4))$. The successive outputs are $(\varepsilon, \phi_{(2)})$, $(1_{(1)}, \varepsilon)$, $(\phi_{(1)}, \varepsilon)$, and $(\varepsilon, 1_{(1)})$, hence the global output is the pair $(1_{(1)} \cdot \phi_{(1)}, \phi_{(2)} \cdot \phi_{(1)})$. According to Lemma 3.5, we thus have found a common LD-expansion of t_1 and t_2 , here $t' = (t_1)1_{(1)} \cdot \phi_{(1)} = (t_2)\phi_{(2)} \cdot \phi_{(1)}$.

When it converges, the Polish Algorithm solves the word problem of (LD) and decides the canonical ordering in the following strong sense:

Proposition 3.6. Assume that the Polish Algorithm running on the pair (t_1, t_2) converges to the pair (t'_1, t'_2) . Then

- (i) $t_1 =_{LD} t_2$ is equivalent to $t'_1 = t'_2$;
- (ii) $t_1 \sqsupset_{LD} t_2$ is equivalent to $t'_1 \sqsupset t'_2$;
- (iii) $t_1 \gg_{LD} t_2$ is equivalent to $t'_1 \gg t'_2$.

Proof. By construction, we have $t'_1 =_{LD} t_1$ and $t'_2 =_{LD} t_2$, so the conditions of the proposition are sufficient. They are also necessary, for the hypothesis that t'_1 and t'_2 have no soft clash implies that exactly one of $t'_1 = t'_2, t'_1 \sqsubset t'_2, t'_1 \sqsupset t'_2, t'_1 \ll t'_2, t'_1 \gg t'_2$ holds. ■

Remark. The Polish algorithm can be extended so as to run on an arbitrary finite sequence of terms (t_1, t_2, \dots) instead of a pair: when a soft clash occurs, we expand using Lemma 3.4 all terms where a variable occurs at the considered position. The details are easy.

Progressive words

If t_1 and t_2 are LD-equivalent terms, then they admit a common LD-expansion. The principle of the Polish Algorithm running on (t_1, t_2) is to try to construct such a common LD-expansion, *i.e.*, to find words u_1, u_2 on \mathbf{A} — *i.e.*, finite sequences of addresses — such that $(t_1)u_1$ and $(t_2)u_2$ coincide. However, the output words provided by the Polish Algorithm cannot be arbitrary words.

Definition. For u a word on \mathbf{A} , we say that u is *progressive* if it admits a decomposition $u = \alpha_1(k_1) \dots \alpha_m(k_m)$ such that $\alpha_i 10^{k_i-1} 1 >_{LR} \alpha_{i-1} 0^{k_{i-1}}$ holds for $1 < i \leq m$. We say that t' is a *progressive LD-expansion* of t if we have $t' = (t)u$ for some progressive word u .

Lemma 3.7. *The output of the Polish Algorithm consists of progressive words.*

Proof. Assume that $\alpha_{(k)}$ and $\beta_{(j)}$ are successive factors in an output word of the Polish Algorithm. This means that there exist three terms t, t', t'' such that $t' = (t)\alpha_{(k)}$ and $t'' = (t')\beta_{(j)}$ hold. By hypothesis, the terms t and t' have a soft clash at some position p , and the terms t' and t'' have a soft clash at some position q . The hypothesis of $\beta_{(j)}$ occurring after $\alpha_{(k)}$ in an output word implies $q > p$: the clashes always go right, and never back. By Lemma 3.3, the address of p in the middle term t' is $\alpha 0^k$, while the address of q in t' is $\beta 10^{j-1} 10^*$. The correspondence between addresses and positions is increasing, hence $q > p$ translates into $\beta 10^{j-1} 10^* >_{LR} \alpha 0^k$, which is equivalent to $\beta 10^{j-1} 1 >_{LR} \alpha 0^k$. ■

Proposition 3.8. *Assume that t_1, t_2 are terms. Then the following are equivalent:*

- (i) *The term t_2 is a progressive LD-expansion of the term t_1 ;*
- (ii) *The Polish Algorithm running on the pair (t_1, t_2) converges to (t_2, t_2) .*

Proof. We prove using induction on m that, if u is a word in \mathbf{A}^* that admits a progressive decomposition of length m , then the Polish Algorithm running on every pair $(t, (t)u)$ returns $((t)u, (t)u)$ with output (ε, u) in m steps. For $m = 0$, u must be empty, and the result is true. Assume $m > 0$. Write $t_e^{(i)}$ for the term obtained after i steps of the algorithm. Then the clash between $t_1^{(0)}$, *i.e.*, t_1 , and $t_1^{(m)}$, *i.e.*, $(t_1)u$, is the clash between $t_1^{(0)}$ and $t_1^{(1)}$, as the subsequent clashes occur further to the right. Hence the Polish Algorithm gives the first factor of u at the first step, and the induction is obvious.

If the Polish Algorithm running on (t_1, t_2) converges to (t, t) , then, by Lemma 3.7, t is a progressive LD-expansion both of t_1 and t_2 . If (ii) holds, we have $t = t_2$, and (i) follows. ■

The Polish Algorithm below a full term

We show now how to deduce convergence results for the Polish Algorithm from the results of Section 2. The idea is that, if t_0 is a full term and every element of FLD_∞ appearing in t_1 and t_2 also appears in t_0 , then every element appearing in every term occurring when the Polish Algorithm is run on the pair (t_1, t_2) appears in t_0 as well.

Definition. Assume that t, t_0 are terms. We say that t is *included* in t_0 if the content of t is included in the content of t_0 , *i.e.*, if every element appearing in t also appears in t_0 , and \bar{t} either appears in t_0 or it is equal to \bar{t}_0 .

Lemma 3.9. *Assume that t is included in t_0 , and that a covers b in t . Then a covers b in t_0 as well.*

Proof. Let a_0 be the least element covered by a in t . Let a^+ be the least element that appears on the right of a in t , or \bar{t} if a is the largest element appearing in t . By hypothesis, a_0 appears in t_0 , and a^+ appears in t_0 or it is \bar{t}_0 . By Lemma 1.14, the elements a^+ and aa_0 are almost equal, *i.e.*, they may differ only by the rightmost variable. In particular, their projections $(a^+)^\dagger$ and $(aa_0)^\dagger$ on FLD_1 are equal. Let b_0 be least element covered by a in t_0 , and b^+ be the least element that appears on the right of a in t_0 , or \bar{t}_0 if a is the largest element appearing in t_0 . We have similarly $(b^+)^\dagger = (ab_0)^\dagger$, and, by construction, $a^+ \supseteq b^+$ holds. Hence, we find,

$$a^\dagger a_0^\dagger = (a^+)^\dagger \supseteq (b^+)^\dagger = a^\dagger b_0^\dagger,$$

which implies $a_0^\dagger \supseteq b_0^\dagger$, and, therefore, $a_0 \supseteq b_0$, since, otherwise, we would get $b_0 \sqsupset a_0$ and $b_0^\dagger \sqsupset a_0^\dagger$. Now b satisfies $a > b \geq a_0$, hence it satisfies $a > b \geq b_0$ as well, and, therefore, it is covered by a in t_0 , since the elements covered by a in t_0 make an interval in the content of t_0 . ■

Lemma 3.10. *Assume that t_0 is a full term, t_1, t_2 are included in t_0 , and the Polish Algorithm running on (t_1, t_2) returns (t'_1, t'_2) in one step. Then t'_1 and t'_2 are included in t_0 .*

Proof. (Figure 3.2) By hypothesis, t_1 and t_2 have a soft clash. Let us assume that the clash occurs at position p , with a variable in \tilde{t}_1 and a letter \bullet in \tilde{t}_2 . Let $\gamma 10^k 1^*$ be the address of p in t_1 , and let a_1, \dots, a_k be the elements that appear in t_1 respectively at the addresses $\gamma 10^k 1^*, \gamma 10^{k-1} 1^*, \dots, \gamma 10 1^*$. Let a_0 be the element that appears at $\gamma 10^*$, and, finally, let b_1, \dots, b_ℓ be the increasing enumeration of those elements that appear in t_1 below $\gamma 0$. By hypothesis, we have $t'_2 = t_2$ and $t'_1 = (t_1)\gamma_{(k)}$. Hence, the elements appearing in t'_1 are those elements appearing in t_1 , completed with the elements $a_i b_j$ with $1 \leq i \leq k$ and $1 \leq j \leq \ell$. Our aim is to prove that the latter elements appear in t_0 .

To this end, we use the covering relations in t_0 , and the hypothesis that t_0 is full. First, b_ℓ covers b_j in t_1 for $j = 1, \dots, \ell - 1$, so, by Lemma 3.9, b_ℓ covers b_j in t_0 as well. Similarly, for $i = 2, \dots, k$, a_i covers a_1 in t_1 , so a_i covers a_1 in t_0 as well. Now, a_1 covers a_0 in t_1 , and, by hypothesis, the words \tilde{t}_1 and \tilde{t}_2 coincide up to position p , and a_1 covers more elements in t_2 than in t_1 . As the immediate predecessor of a_0 in t_1 and t_2 is b_ℓ , a_1 must cover b_ℓ in t_2 . This implies that a_1 covers b_ℓ in t_0 . As the covering relation is transitive, we deduce that a_i covers b_j in t_0 for $i = 1, \dots, k$ and $j = 1, \dots, \ell$. Now, the elements $a_i b_j$ appear in t'_1 , which is an LD-expansion of t_1 , so we must have $\tilde{t}_1 \sqsupseteq a_i b_j$, and, therefore $\tilde{t}_0 \sqsupseteq a_i b_j$. Hence $a_i b_j$ appears in some LD-expansion of t_0 . Now a_i covers b_j in t_0 , and t_0 is full, so the hypothesis that $a_i b_j$ appears in some LD-expansion of t_0 implies that $a_i b_j$ already appears in t_0 . ■

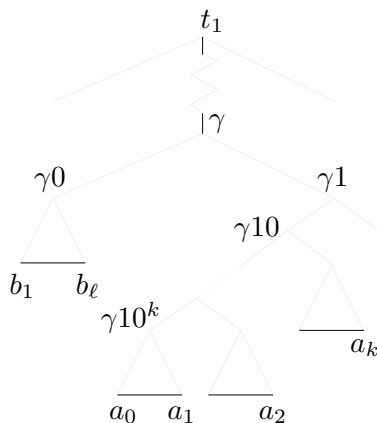


Figure 3.2. Convergence of the Polish Algorithm

We deduce immediately:

Proposition 3.11. *Assume that t_0 is a full term, and t_1, t_2 are included in t_0 . Then the Polish Algorithm running on (t_1, t_2) converges in a finite number of steps bounded by the size of t_0 .*

Corollary 3.12. *Assume that t, t_0 are terms and t_0 is full. The following are equivalent:*

- (i) *The term t is included in t_0 ;*
- (ii) *Some cut of t_0 is an LD-expansion of t ;*
- (iii) *Some cut of t_0 is a progressive LD-expansion of t .*

Proof. For every term t_0 , (ii) implies (i), and (iii) implies (ii). Now, assume that t_0 is full and t is included in t_0 . Proposition 3.11 tells us that the Polish Algorithm running on (t, t_0) converges to a pair of the form (t', t_0) . The hypothesis that t' and t_0 have no soft clash together with $\tilde{t} \sqsubseteq \tilde{t}_0$ implies $t' \sqsubseteq t_0$, so t' is a cut of t_0 . ■

The previous results imply the convergence of the Polish Algorithm for every pair of initial terms (t_1, t_2) such that t_1 and t_2 are included in a full term, hence, in particular, if a term of the form $\partial^k t_0$ with t_0 a (quasi)-injective term. Actually, we can do better by considering the following notion of a lifting.

Definition. Assume that (t_1, t_2, \dots) is a sequence of terms in T_∞ . We say that the sequence (t'_1, t'_2, \dots) is a *lifting* of (t_1, t_2, \dots) if there exists a mapping f of $\{x_1, x_2, \dots\}$ into itself such that $t_i = (t'_i)^f$ holds for each i .

Thus a lifting of a term t is obtained from t by changing the names of the variables with the only constraint that distinct variables cannot be replaced with the same variable. We do not require the replacement to be functional: different occurrences of the same variable may be replaced with different variables. So, for instance, every term is a lifting of a one-variable term, and every term admits an injective lifting.

Lemma 3.13. Assume that (t'_1, t'_2) is a lifting of (t_1, t_2) .

(i) If the Polish Algorithm running on (t_1, t_2) converges, then it converges as well when running on (t'_1, t'_2) .

(ii) Conversely, if t'_1 and t'_2 are \sqsubseteq_{LD} -comparable, and the Polish Algorithm running on (t'_1, t'_2) converges, then it converges as well when running on (t_1, t_2) .

Proof. Let us compare how the Polish Algorithm runs on (t_1, t_2) and on (t'_1, t'_2) . Assume that the algorithm converges for (t_1, t_2) in m steps. We claim that it converges for (t'_1, t'_2) in at most m steps. For an induction, it suffices to consider the first step. Assume that t_1 and t_2 have a clash at position p . Then, either t'_1 and t'_2 have a variable clash at some position $< p$, in which case the Polish Algorithm halts immediately, or they coincide at least until position $p - 1$. If the clash of t_1 and t_2 at p is strong, then t'_1 and t'_2 have a strong clash of the same type at p : indeed, different variables in t_1 and t_2 must come from distinct variables in t'_1 and t'_2 . If the clash of t_1 and t_2 at p is soft, t'_1 and t'_2 also have a soft clash at p . In this case, the pair $((t'_1)^{(1)}, (t'_2)^{(1)})$ obtained when treating the clash of t'_1 and t'_2 at p is a lifting of the pair $(t_1^{(1)}, t_2^{(1)})$ obtained when treating the clash of t_1 and t_2 , and the induction goes on. This proves (i).

For (ii), we prove similarly that, if the algorithm converges for (t'_1, t'_2) in m steps, then it converges for (t_1, t_2) in m steps too. The argument is the same as above. The only problem is that t'_1 and t'_2 may have a variable clash that t_1 and t_2 do not have, as the mapping involved in the lifting need not be injective. Now the hypothesis that t'_1 and t'_2 are \sqsubseteq_{LD} -comparable discards this possibility. ■

Proposition 3.14. Assume that (t_1, t_2) is a pair of terms admitting some lifting (t'_1, t'_2) such that $t_0 \sqsupseteq_{LD} t'_1$ and $t_0 \sqsupseteq_{LD} t'_2$ holds for some quasi-injective term t_0 . Then the Polish Algorithm converges when running on (t_1, t_2) .

Proof. For k large enough, the terms t'_1 and t'_2 are included in $\partial^k t_0$, which is full. Hence the Polish algorithm for (t'_1, t'_2) converges. By Lemma 3.13, this implies that it converges for (t_1, t_2) as well. ■

Example 3.15. Let us consider the case of one variable terms. The previous criterion applies to all pairs (t_1, t_2) for which we can find a lifting, *i.e.*, a renaming of the variables, providing terms that are \sqsubseteq_{LD} -comparable to some common quasi-injective term. If only one term were involved, we could consider an arbitrary injective lifting. The point is that, if we rename the variables of t_1 say x_1, x_2, \dots , there need not exist a lifting of t_2 giving a

\sqsubseteq_{LD} -comparable term. The smallest pair for which the criterion of Proposition 3.14 does not apply is the pair

$$t_1 = \begin{array}{c} \diagup \quad \diagdown \\ x \quad x \\ \diagdown \quad \diagup \\ x \quad x \end{array}, \quad t_2 = \begin{array}{c} \diagup \quad \diagdown \\ x \quad x \\ \diagdown \quad \diagup \\ x \quad x \\ \diagdown \quad \diagup \\ x \quad x \end{array}.$$

Up to a permutation, the only injective lifting of t_1 is $t'_1 = x_1x_2x_3x_4$, but there is no lifting of t_2 that makes it \sqsubseteq_{LD} -comparable with t'_1 . The reason is as follows: let t_3 be the LD-expansion $(t_2)1$. Then t_3 admits a lifting making it \sqsubseteq_{LD} -comparable with t'_1 , namely

$$t'_3 = \begin{array}{c} \diagup \quad \diagdown \\ x_1 \quad x_1 \\ \diagdown \quad \diagup \\ x_1 \quad x_2 \quad x_3 \quad x_4 \end{array}.$$

But this lifting cannot be carried to t_2 because of the clash between the variables x_1 and x_3 at 100 and 110.

Proposition 3.16. *Assume that t_1 and t_2 are LD-expansions of \sqsubseteq -comparable terms. Then the Polish Algorithm converges when running on (t_1, t_2) .*

Proof. Assume $t_e = (s_e)u_e$ for $e = 1, 2$, where u_e is a word in \mathbf{A}^* , and $s_1 \sqsubseteq s_2$ holds for instance. Let s'_2 be an injective lifting of s_2 , s'_1 be the induced lifting of s_1 , and t'_e be the term $(s'_e)u_e$, which exists as u_e is positive. Then (t'_1, t'_2) is a lifting of (t_1, t_2) , and Proposition 3.14 applies. \blacksquare

The criterion of Proposition 3.16 shows in particular that the Polish Algorithm converges on every pair (t_1, t_2) such that t_1 and t_2 both are LD-expansions of some term t_0 .

Degree k expansions

Here we consider the particular case of the terms $\partial^k t$.

Proposition 3.17. *Let t_1, t_2 be arbitrary terms. Then, for k large enough, the Polish Algorithm converges when running on $(\partial^k t_1, t_2)$ and on $(\partial^k t_1, \partial^k t_2)$.*

Proof. Let t_1^\dagger and t_2^\dagger be the projections of t_1 and t_2 on T_1 . Choose m such that $t_2^\dagger \sqsubseteq_{LD} t_1^\dagger x^{[m]}$ holds [6]. For k large enough, some cut of $\partial^k(t_1^\dagger x^{[m]})$ is an LD-expansion of t_2^\dagger . By Proposition 3.16, the Polish Algorithm converges for $(\partial^k(t_1^\dagger x^{[m]}), t_2^\dagger)$, hence for $(\partial^k t_1^\dagger, t_2^\dagger)$ since $\partial^k t_1$ is a cut of $\partial^k(t_1^\dagger x^{[m]})$. By construction, $(\partial^k t_1, t_2)$ is a lifting of $(\partial^k t_1^\dagger, t_2^\dagger)$, so Lemma 3.13 implies that the Polish Algorithm converges for $(\partial^k t_1, t_2)$ as well.

On the other hand, it is known [7] that some term t^\dagger admits two iterated left subterms s_1, s_2 that are LD-expansions of t_1^\dagger and t_2^\dagger respectively. For k large enough, the term $\partial^k t_e^\dagger$ is an LD-expansion of s_e , and, by Proposition 3.16 again, the Polish algorithm converges for $(\partial^k t_1^\dagger, \partial^k t_2^\dagger)$, hence for $(\partial^k t_1, \partial^k t_2)$. \blacksquare

Observe that the previous result gives an always terminating process for comparing terms using the Polish Algorithm: it suffices to run the latter on the pairs $(t_1, t_2), (\partial t_1, \partial t_2), \dots$ simultaneously—perform m steps of the algorithm on the pairs $(t_1, t_2), \dots, (\partial^m t_1, \partial^m t_2)$ for $m = 0, 1, 2, \dots$ so as to obtain an effective process.

We also deduce from Proposition 3.17 that, for all LD-equivalent terms t_1, t_2 , the terms $\partial^k t_1$ and $\partial^k t_2$ admit a common progressive LD-expansion for k large enough.

Definition. Assume that t and t' are terms. We say that t' is a *degree k LD-expansion* of t if t' is an LD-expansion of t and $\partial^k t$ is an expansion of t' , but $\partial^{k-1} t$ is not. We write $\text{Exp}_k(t)$ for the set of all LD-expansions of t with degree at most k , and $\text{Exp}_k^{\text{prog}}(t)$ for the set of all progressive LD-expansions of t with degree at most k .

We know that, if t' is an LD-expansion of t , then $\partial^k t$ is an LD-expansion of t' for every k large enough. So every LD-expansion has a well defined finite degree.

Proposition 3.18. *Assume that the term t' is an LD-expansion of the term t . Then the following are equivalent:*

- (i) *The term t' is included in $\partial^k t$;*
- (ii) *The term t' is an LD-expansion of t of degree at most k , i.e., t' belongs to $\text{Exp}_k(t)$;*
- (iii) *The term $\partial^k t$ is a progressive LD-expansion of t' , i.e., $\partial^k t$ belongs to $\text{Exp}_k^{\text{prog}}(t')$.*

Proof. Let t' be an injective lifting of t , say $t = (t')^f$. Then the term $\partial^k t'$ is full, and Proposition 3.14 gives the equivalence of (i), (ii), and (iii) for the LD-expansions of t' . Now, for each LD-expansion t' of t , there exists a unique LD-expansion t'' of t' with the same outline as t' . Then we have $t' = (t'')^f$, and each of (i), (ii), (iii) for t' is equivalent to its counterpart for t'' . ■

Proposition 3.19. *For every term t , and every k , $\text{Exp}_k(t)$ is closed under Polish Algorithm in the sense that, for t_1, t_2 in $\text{Exp}_k(t)$, the Polish Algorithm running on (t_1, t_2) returns a common (progressive) LD-expansion of t_1 and t_2 that lies in $\text{Exp}_k(t)$.*

Proof. As above, we use an injective lifting t' of t . For (t'_1, t'_2) , the result follows from Proposition 3.11. We then deduce the result for (t_1, t_2) using a projection. ■

Proposition 3.20. *Assume that t' is an LD-expansion of t . Then $\partial^k t'$ is a progressive LD-expansion of $\partial^k t$ for k large enough.*

Proof. We prove the general result that, for $t' \in \text{Exp}_k(t)$ and $n \geq 1$, we have $\partial^{nk} t \in \text{Exp}_k(\partial^{(n-1)k} t')$ and $\partial^{nk} t' \in \text{Exp}_k(\partial^{nk} t)$. We use induction on $n \geq 1$. Assume first $n = 1$. The hypothesis is that $\partial^k t$ is an LD-expansion of t . By Proposition 3.18, $\partial^k t$ is a progressive LD-expansion of t' . Now, as t' is an LD-expansion of t , $\partial^k t'$ is an LD-expansion of $\partial^k t$, hence the LD-class of every cut of $\partial^k t$ appears in $\partial^k t'$, i.e., $\partial^k t$ is included in $\partial^k t'$. By Proposition 3.18 again, $\partial^k t$ is an LD-expansion of t' of degree at most k .

Applying the previous argument to the pair $(t', \partial^k t)$ instead of (t, t') shows that $\partial^k t'$ is a progressive LD-expansion of $\partial^k t$ of degree at most k .

Assume now $n \geq 2$. By induction hypothesis, $\partial^{(n-1)k} t'$ belongs to $\text{Exp}_k(\partial^{(n-1)k} t)$. Hence, by the above argument, the term $\partial^k(\partial^{(n-1)k} t)$, i.e., $\partial^{nk} t$, is a progressive LD-expansion of $\partial^{(n-1)k} t'$ of degree k at most. Always by the same argument, the term $\partial^k(\partial^{(n-1)k} t')$, i.e., $\partial^{nk} t'$, is a progressive LD-expansion of $\partial^{nk} t$ of degree k at most. ■

4. THE TABLE OF A TERM

A unique normal form for LD-equivalence has been constructed in [7] : for every fixed term t_0 , there exists a family of so-called t_0 -normal terms such that every term t satisfying $t \sqsubseteq_{LD} t_0$ is LD-equivalent to a unique t_0 -normal term. The computation of the t_0 -normal form has a primitive recursive complexity, but, in practice, the method of [7] is intractable beyond very simple terms. As application for the results for Section 2, we develop now a new, simpler method for computing the t_0 -normal form when t_0 is a quasi-injective term, or, more generally, a perfect term.

The right normal form

Here we recall (without proof) the normal form result of [7]. It is stated here in a new, hopefully more accessible form, relying on the notion of a fractional cut.

By Proposition 2.5, every term t satisfying $t \sqsubseteq_{LD} t_0$ is LD-equivalent to some cut of $\partial^k t_0$ for k large enough. Hence, it suffices to construct normal forms for the cuts of $\partial^k t_0$. Now, by Formula (1.3), each cut of $\partial^k t_0$ admits, up to LD-equivalence, a unique decomposition as a decreasing product of cuts of $\partial^{k-1} t_0$. Thus, every cut of $\partial^k t_0$ can be specified by a sequence of cuts of $\partial^{k-1} t_0$, hence a sequence of sequences of cuts of $\partial^{k-2} t_0, \dots$, and, finally, a tree whose leaves are cuts of t_0 .

Definition. For $1 \leq n \leq \infty$, an n -precode is defined to be a term on $\{1, \dots, n\}$, i.e., a finite binary tree with leaves wearing labels in $\{1, \dots, n\}$. For t a term in T_∞ of size n , and \mathbf{c} an n -precode, the *fractional cut* $\text{cut}_{\mathbf{c}}(t)$ is defined inductively by the rules: for $\mathbf{c} = i \in \{1, \dots, n\}$, $\text{cut}_{\mathbf{c}}(t_0)$ is the cut of t_0 ending at the i -th variable from the left; for $\mathbf{c} = \mathbf{c}_1 \mathbf{c}_2$, we have $\text{cut}_{\mathbf{c}}(t) = \text{cut}_{\mathbf{c}_1}(t_0) \text{cut}_{\mathbf{c}_2}(t_0)$.

Thus 2, 3·1, or (3·(2·1))·1 are typical precodes. It is convenient to represent precodes using rational numbers. For \mathbf{c} an n -precode, we define the base n expansion of \mathbf{c} as the word obtained from the right Polish form of \mathbf{c} by replacing the operator \bullet with the digit 0, removing the final 0's, and adding a dot after the first digit. For instance, the expansion of the above precodes are 2, 3.1, and 3.21001 respectively, and, for t_0 a term of size 3 at least, say $t_0 = x_1 x_2 x_3$, we have

$$\text{cut}_2(t_0) = x_1 x_2, \quad \text{cut}_{3.1}(t_0) = (x_1 x_2 x_3) x_1, \quad \text{cut}_{3.21001}(t_0) = ((x_1 x_2 x_3)(x_1 x_2) x_1) x_1.$$

Every rational with a finite base n expansion represents exactly one precode, because the right Polish form is non-ambiguous. The interest of our denotational convention lies that the standard ordering of the rationals corresponds to the order needed in the subsequent construction (simply denoted $<$ in the sequel); in particular, it makes it intuitive that the fractional cut $\text{cut}_{2.1}(t_0)$ lies between $\text{cut}_2(t_0)$ and $\text{cut}_3(t_0)$.

By the initial remark, every cut of $\partial^k t_0$ is LD-equivalent to some fractional cut of t_0 , and it is natural to use precodes to construct a normal form.

Definition. (i) For \mathbf{c} a precode, the *degree* $\text{deg}(\mathbf{c})$ is defined inductively by $\text{deg}(\mathbf{c}) = 0$ for \mathbf{c} an integer, and $\text{deg}(\mathbf{c}) = \sup(\text{deg}(\mathbf{c}_1) + 1, \text{deg}(\mathbf{c}_2))$ for $\mathbf{c} = \mathbf{c}_1 \cdot \mathbf{c}_2$.

(ii) Assume that t_0 is a term of size n , and \mathbf{c} is an n -precode; the t_0 -*head* $\text{hd}_{t_0}(\mathbf{c})$ of \mathbf{c} is defined inductively by

- $\text{hd}_{t_0}(\mathbf{c}) = j$ if \mathbf{c} is the integer i and $\text{cut}_j(t_0)$ is the least cut of t_0 covered by $\text{cut}_i(t_0)$, if any, or $j = i$ otherwise,
- $\text{hd}_{t_0}(\mathbf{c}) = \mathbf{c}_1 \cdot \text{hd}_{t_0}(\mathbf{c}_2)$ if $\mathbf{c} = \mathbf{c}_1 \cdot \mathbf{c}_2$ holds with $\text{deg}(\mathbf{c}_2) > \text{deg}(\mathbf{c}_1)$,
- $\text{hd}_{t_0}(\mathbf{c}) = \mathbf{c}_1 \cdot 1$ if $\mathbf{c} = \mathbf{c}_1 \cdot \mathbf{c}_2$ holds with $\text{deg}(\mathbf{c}_2) \leq \text{deg}(\mathbf{c}_1)$.

Thus the degree of \mathbf{c} is the maximal number of 0's in an address in \mathbf{c} (viewed as a binary tree) and the t_0 -head of \mathbf{c} is obtained from \mathbf{c} by keeping the complicated left part of \mathbf{c} , but deleting the remaining fragment and replacing it with 1. For instance, the degree of 3.2 is 1, and its $x_1 x_2 x_3 x_4$ -head is 3.1, while the degree of 3.210032 is 2, and its $x_1 x_2 x_3 x_4$ -head is 3.21001.

Definition. Assume that t_0 is a term of size n . We say that a precode \mathbf{c} is a t_0 -*code* if \mathbf{c} is an n -precode and, for every internal address α in \mathbf{c} , we have

$$\text{deg}(\text{sub}(\mathbf{c}, \alpha 0)) + 1 \geq \text{deg}(\text{sub}(\mathbf{c}, \alpha 1)), \quad \text{and} \quad \text{hd}_{t_0}(\text{sub}(\mathbf{c}, \alpha 0)) > \text{sub}(\mathbf{c}, \alpha 1). \quad (4.1)$$

Example 4.1. Let $t_0 = x^{[n]}$ (we write x for x_1). The t_0 -codes of degree 0 are the integers $1, \dots, n$. A degree 1 precode is a finite sequence of integers, say (i_1, \dots, i_{r+1}) . For every i , we have $\text{hd}_{t_0}(i) = i$ here, hence Condition (4.1) reduces to $i_p > i_{p+1}$. So the t_0 -codes of degree 1 are the decreasing sequences (i_1, \dots, i_{r+1}) with $n \geq i_1$. Thus the first t_0 -codes of degree 1 are 2.1, 3.1, 3.21, 4.1, 4.2, 4.21, 4.3, \dots

Things become more complicated for 2-codes. By definition, a t_0 -code of degree 2 is a decreasing sequence of t_0 -codes of degree 1, but the condition is no longer sufficient. For instance, the precode 3.2031, *i.e.*, $(3 \cdot 2) \cdot (3 \cdot 1)$, is not a t_0 -code. Indeed, the degree condition holds, but the t_0 -head of $\text{sub}(3.2031, 0)$, *i.e.*, of 3.2, is 3.1, which is not larger than $\text{sub}(3.2031, 1)$, which is 3.1 as well.

In the current framework, the normal form result of [7] can be restated as follows:

Proposition 4.2. *Assume that t_0 is a term. Then, for every term satisfying $t \sqsubseteq_{LD} t_0$, there exists a unique t_0 -code \mathbf{c} satisfying $t =_{LD} \text{cut}_{\mathbf{c}}(t_0)$. The function that maps t to \mathbf{c} lies in the complexity class $\text{DSPACE}(\exp^*(\mathcal{O}(2^n)))$, where $\exp^*(m)$ denotes a tower of base 2 exponentials of height m . For $t_1, t_2 \sqsubseteq_{LD} t_0$, $t_1 \sqsubset_{LD} t_2$ is equivalent to $\mathbf{c}_1 < \mathbf{c}_2$, where \mathbf{c}_e is the t_0 -code associated with t_e as above.*

If $t'_0 \sqsupset t_0$ holds, every t_0 -code is a t'_0 -code. Using this remark, we can extend the notion of a t_0 -code to some infinite terms t_0 . In this way, we obtain for instance that every term in T_1 is associated with a unique x^∞ -code, where x^∞ represents the limit of $x^{[n]}$ when n goes to ∞ .

The table of a term

Assume that t_0 is a term in T_∞ . By the previous results, every element of FLD_∞ that appears in some LD-expansion of t_0 is specified by a unique t_0 -code. The question of describing the pairs (a, b) of elements appearing in an LD-expansion of t_0 such that ab still appears in an LD-expansion of t_0 can be restated as the question of finding, for every pair $(\mathbf{c}_1, \mathbf{c}_2)$ of t_0 -codes, the unique t_0 -code \mathbf{c} satisfying $\text{cut}_{\mathbf{c}}(t_0) =_{LD} \text{cut}_{\mathbf{c}_1}(t_0)\text{cut}_{\mathbf{c}_2}(t_0)$, if it exists. This leads us to the following notion.

Definition. Assume that t_0 is a term. The k -table of t_0 is defined to be the pair $(C_k^{t_0}, *_k^{t_0})$, where $C_k^{t_0}$ is the set of all t_0 -codes of degree $\leq k$ except the last one, and $*_k^{t_0}$ is the partial binary operation on $C_k^{t_0}$ defined by

$$\mathbf{c}_1 *_k^{t_0} \mathbf{c}_2 = \begin{cases} \mathbf{c} & \text{if } \text{cut}_{\mathbf{c}_1}(t_0)\text{cut}_{\mathbf{c}_2}(t_0) \text{ is LD-equivalent to } \text{cut}_{\mathbf{c}}(t_0), \\ \perp & \text{if } \text{cut}_{\mathbf{c}_1}(t_0)\text{cut}_{\mathbf{c}_2}(t_0) \text{ appears in no LD-expansion of } t_0. \end{cases}$$

The k -table of t_0 corresponds to the fragment of the multiplication table of FLD_∞ restricted to the cuts of $\partial^k t_0$. By construction, $\mathbf{c}_1 *_k^{t_0} \mathbf{c}_2$ is undefined if $\text{cut}_{\mathbf{c}_1}(t_0)\text{cut}_{\mathbf{c}_2}(t_0)$ appears in some LD-expansion of $\partial^k t_0$, but not in $\partial^k t_0$, *i.e.*, when it is associated with a code of degree higher than k . On the other hand, \perp indicates a permanent obstruction. Observe that every k -table is a partial left self-distributive table, in the sense that, if the values of $a * (b * c)$ and $(a * b) * (a * c)$ both exist, then they are equal.

The results of Section 2 enable us to describe the table of a quasi-injective term completely. In the sequel, we do not distinguish between the t_0 -codes and the associated fractional cuts of t_0 . So we can speak of the product of two codes, or of the decomposition of a degree k code as a sequence of degree $k - 1$ codes. For finite sequences of degree k t_0 -codes, we define $*_k^{t_0}$ -reduction as in Section 2, but replacing the product of FLD_∞ with its counterpart $*_k^{t_0}$ on $C_k^{t_0}$.

Proposition 4.3. *Assume that t_0 is a quasi-injective term of size n .*

(i) *The set $C_0^{t_0}$ is $\{1, \dots, n-1\}$, and the operation $*_0^{t_0}$ is defined by $\mathbf{c}_1 *_0^{t_0} \mathbf{c}_2 = \perp$ for $\text{cut}_{\mathbf{c}_1}(t_0)$ covering $\text{cut}_{\mathbf{c}_2}(t_0)$ in t_0 , and undefined otherwise.*

(ii) *For $k > 0$, the k -table of t_0 is obtained from the $(k-1)$ -table of t_0 as follows:*

(ii₁) *The set $C_k^{t_0}$ is the set of all codes $\mathbf{c}_1 \dots \mathbf{c}_{p+1}$ such that, for every i , \mathbf{c}_i belongs to $C_{k-1}^{t_0}$ and $\mathbf{c}_i *_k^{t_0} \mathbf{c}_{i+1}$ is undefined;*

(ii₂) *For $\mathbf{c}_1, \mathbf{c}_2$ in $C_k^{t_0}$ satisfying $\mathbf{c}_1 \leq \mathbf{c}_2$, we have $\mathbf{c}_1 *_k^{t_0} \mathbf{c}_2 = \perp$;*

(ii₃) *For $\mathbf{c}_1, \mathbf{c}_2$ in $C_k^{t_0}$ satisfying $\mathbf{c}_1 > \mathbf{c}_2$ and of the form*

$$\mathbf{c}_1 = \mathbf{c}'_1 \dots \mathbf{c}'_p \mathbf{c}'_{p+1} \quad \text{and} \quad \mathbf{c}_2 = \mathbf{c}'_1 \dots \mathbf{c}'_p \mathbf{c}''_{p+1} \dots \mathbf{c}''_{q+1}$$

*with $\mathbf{c}'_{p+1} > \mathbf{c}''_{p+1}$, we have $\mathbf{c}_1 *_k^{t_0} \mathbf{c}_2 = \mathbf{c}'''_1 \dots \mathbf{c}'''_r$, where $(\mathbf{c}'''_1, \dots, \mathbf{c}'''_r)$ is the irreducible sequence obtained from $(\mathbf{c}'_1, \dots, \mathbf{c}'_p, \mathbf{c}'_{p+1}, \mathbf{c}''_{p+1}, \dots, \mathbf{c}''_{q+1})$ by using $*_{k-1}^{t_0}$ -reduction, if this code belongs to $C_k^{t_0}$, and we have $\mathbf{c}_1 *_k^{t_0} \mathbf{c}_2 = \perp$ otherwise;*

(ii₄) *For $\mathbf{c}_1, \mathbf{c}_2$ in $C_k^{t_0}$ satisfying $\mathbf{c}_1 > \mathbf{c}_2$ but not of the form (ii₃), we leave $\mathbf{c}_1 *_k^{t_0} \mathbf{c}_2$ undefined.*

Proof. Without loss of generality, we may assume that t_0 is quasi-increasing. Using induction on k , we prove the result of the proposition together with the additional facts that $\mathbf{c}_1 *_k^{t_0} \mathbf{c}_2$ is undefined if and only if $\text{cut}_{\mathbf{c}_1}(t_0)$ uncovers $\text{cut}_{\mathbf{c}_2}(t_0)$ in $\partial^k t$, and $\mathbf{c}_1 *_k^{t_0} \mathbf{c}_2 = \perp$ holds if and only if $\text{cut}_{\mathbf{c}_1}(t_0)$ covers $\text{cut}_{\mathbf{c}_2}(t_0)$ in $\partial^k t$ and $t_0 \not\equiv_{LD} \text{cut}_{\mathbf{c}_1}(t_0) \cdot \text{cut}_{\mathbf{c}_2}(t_0)$ does not hold. For $k = 0$, everything is true as no cut of a quasi-increasing term may be LD-equivalent to the product of two other cuts. Then induction follows from Lemma 2.18, which, in addition to proving that $\partial^k t$ is perfect, also gives the effective computation method by a reduction. ■

Example 4.4. Let $t_0 = x_1 x_2 x_3 x_4$. Then t_0 is injective, hence eligible for the previous result—while its substitute $x^{[4]}$ is not. The 0-table and the 1-table are

$*_t^0$	1	2	3	$*_t^1$	1	2	2.1	3	3.1	3.2	3.21
1	⊥	⊥	⊥	1	⊥	⊥	⊥	⊥	⊥	⊥	⊥
2	.	⊥	⊥	2	2.1	⊥	⊥	⊥	⊥	⊥	⊥
3	.	.	⊥	2.1	.	.	⊥	⊥	⊥	⊥	⊥
				3	3.1	3.2	3.21	⊥	⊥	⊥	⊥
				3.1	⊥	⊥	⊥
				3.2	3.21	⊥	⊥
				3.21	⊥

(No 0-reduction is possible.) Forty-one elements appear in $\partial^2 t$, and writing the complete 2-table of t_0 is tedious. We give below the trace of the 2-table of t_0 on $C_1^{t_0}$, *i.e.*, we fill the empty entries of the previous array.

$*_2^{t_0}$ (excerpt)	1	2	2.1	3	3.1	3.2	3.21
1	⊥	⊥	⊥	⊥	⊥	⊥	⊥
2	2.1	⊥	⊥	⊥	⊥	⊥	⊥
2.1	2.101	2.102	⊥	⊥	⊥	⊥	⊥
3	3.1	3.2	3.21	⊥	⊥	⊥	⊥
3.1	3.101	3.102	3.1021	3.103	⊥	⊥	⊥
3.2	3.201	3.202	3.2021	3.203	3.21	⊥	⊥
3.21	3.2101	3.2102	3.21021	3.2103	3.21031	3.21032	⊥

Actually, the part above is the trivial part of the 2-table. More interesting phenomena appear in the degree 2 part, when $*_1^{t_0}$ -reduction is possible. Let us only consider an example. As $3.2 *_1^{t_0} 3$ is undefined, 3.203 belongs to $C_2^{t_0}$, and so does 3.201 for the same reason. Let us compute the value of $3.203 *_2^{t_0} 3.201$. We have $3.203 > 3.201$, so we are in case (ii₃) or (ii₄) of Proposition 4.3. The 1-decompositions are $(3.2, 3)$ and $(3.2, 1)$ respectively, so we are in case (ii₃), and we have to reduce the sequence $(3.2, 3, 1)$. We see in the table that $3.2 *_1^{t_0} 2$ is not defined, but we find $3 *_1^{t_0} 1 = 3.1$. Hence $(3.2, 3, 1)$ reduces to $(3.2, 3.1)$. Now, we find $3.2 *_1^{t_0} 3.1 = 3.21$. So $(3.2, 3.1)$ reduces to (3.21) . The code 3.21 belongs to $C_2^{t_0}$, and we deduce $3.203 *_2^{t_0} 3.201 = 3.21$.

We invite the reader to compute other values, for instance $3.203 *_2^{t_0} 3.2021 = \perp$.

Computation of the normal form

Computing the t_0 -code of a term is connected with computing the table of t_0 . Indeed, assume $t \sqsubseteq_{LD} t_0$. Then the t_0 -code of t exists. Assume $t = t_1 t_2$. Then $t_1 \sqsubseteq_{LD} t_0$ holds by definition, and, therefore the t_0 -code of t_1 exists. This need not be the case for t_2 , but, if $t_2 \sqsubseteq_{LD} t_0$ happens to hold, then, assuming that we have found the t_0 -codes say \mathbf{c}_1 and \mathbf{c}_2 of t_1 and t_2 , finding the t_0 -code of t amounts to computing $\mathbf{c}_1 *_1^{t_0} \mathbf{c}_2$, which can be done using Proposition 4.3 if t_0 is quasi-injective. The trivial example $t = t_0 = x_1 x_2$ shows that the previous approach may fail even for simple terms. However, the following observation makes it useful.

Lemma 4.5. *Assume that t'_0 is a lifting of the term t_0 , say $t_0 = (t'_0)^f$.*

- (i) *Assume $t \sqsubseteq_{LD} t'_0$. Then $t^f \sqsubseteq_{LD} t_0$ holds, and the t_0 -code of t^f is the t'_0 -code of t .*
- (ii) *The table of t'_0 is included in the table of t_0 .*

Proof. (i) For each k , we have $\partial^k t_0 = (\partial^k t'_0)^f$, and the outlines of the term $\partial^k t_0$ and $\partial^k t'_0$ coincide, hence $\partial^k t_0$ and $\partial^k t'_0$ have the same descents. The construction of the t_0 -code involves the geometry of the terms $\partial^k t_0$ only, whatever the names of the variables are, so an induction shows that, for every k and every cut t of t'_0 , the t_0 -code of a cut t^f of $\partial^k t_0$ is the t'_0 -code of t .

(ii) The relation $\mathbf{c}_1 *_1^{t'_0} \mathbf{c}_2 = \mathbf{c}$ is equivalent to $\text{cut}_{\mathbf{c}_1}(t'_0) \text{cut}_{\mathbf{c}_2}(t'_0) =_{LD} \text{cut}_{\mathbf{c}}(t'_0)$. If this holds, we have $\text{cut}_{\mathbf{c}_1}(t_0) \text{cut}_{\mathbf{c}_2}(t_0) =_{LD} \text{cut}_{\mathbf{c}}(t_0)$, which in turn is equivalent to $\mathbf{c}_1 *_1^{t_0} \mathbf{c}_2 = \mathbf{c}$. ■

Definition. Assume that t_0 and t are terms. The t_0 -precode of t is the precode \mathbf{c} possibly defined by the following rule: if t is a cut of t_0 , say $t = \text{cut}_i(t_0)$, then $\mathbf{c} = i$; otherwise, if $t = t_1 t_2$ holds and the t_0 -precodes \mathbf{c}_1 and \mathbf{c}_2 of t_1 and t_2 respectively exist, we have $\mathbf{c} = \mathbf{c}_1 \cdot \mathbf{c}_2$.

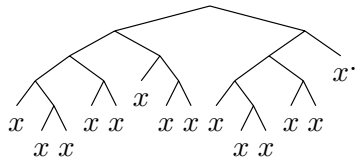
The x^∞ -precode of every one-variable term exists. On the other hand, because of possible variable clashes, the t_0 -precode of t need not always exist when t involves more than one variable.

Assuming that $*$ is a partial binary operation defined on codes, we define the $*$ -evaluation of a code inductively: if \mathbf{c} is an integer, then it is its own evaluation; otherwise, for $\mathbf{c} = \mathbf{c}_1 \mathbf{c}_2$, the $*$ -evaluation of \mathbf{c} is $\mathbf{c}'_1 *_1 \mathbf{c}'_2$, where \mathbf{c}'_e is the $*$ -evaluation of \mathbf{c}_e , when it exists. We deduce from the above result the following practical method for computing the normal form of a term.

Proposition 4.6. *Assume that t_0, t are terms, and $t \sqsubseteq_{LD} t_0$ holds. Let t'_0 be a quasi-injective lifting of t_0 . Then the t_0 -code of t is the evaluation of the t_0 -precode of t in the table of t'_0 , when the latter exists.*

Proof. Assume that the t_0 -precode \mathbf{c} of t exists, and that the evaluation \mathbf{c}' of \mathbf{c} in the table of t'_0 exists as well. By definition, we have $\text{cut}_{\mathbf{c}'}(t'_0) =_{LD} \text{cut}_{\mathbf{c}}(t'_0)$, hence, assuming $t = (t'_0)^f$, $t = \text{cut}_{\mathbf{c}}(t_0) =_{LD} \text{cut}_{\mathbf{c}'}(t_0)$. Now, by construction, \mathbf{c}' is a t'_0 -code, hence a t_0 -code as well by Lemma 4.5. By uniqueness of the t_0 -code, we deduce that \mathbf{c}' is the t_0 -code of t . ■

Example 4.7. Let t be the term



The $x^{[4]}$ -precode of t exists, and it is 3.20303201. We look for the evaluation of this code in the table of the injective lifting $x_1x_2x_3x_4$ of $x^{[4]}$. Using the values computed above, we find

$$((3 * 2) * 3) * ((3 * 2) * 1) = (3.2 * 3) * (3.2 * 1) = 3.203 * 3.201 = 3.21,$$

and we deduce that the $x^{[4]}$ -code of t is 3.21.

When it applies, the method of Proposition 4.6 is much more efficient than the one of Proposition 4.2 as, if we start with a term whose t_0 -precode has degree k , then the entire computation remains in the k -table of t'_0 , for which we have a complete inductive construction. We cannot hope to extend the method to an arbitrary base term t_0 . For instance, no complete description of the 2-table of $x^{[m]}$ is known, and the examples of [11] suggest that such a description must be very complicated for $m \geq 6$.

5. THE EMBEDDING CONJECTURE

In this last section, we come back to the structure of LD-equivalence classes, and, more precisely, to the question of whether any two LD-equivalent terms admit a least common LD-expansion. The question is connected with the presentation of a certain monoid \mathcal{G}_{LD}^+ that describes the action of left self-distributivity on terms, and the latter monoid is related with an extension G_{LD} of Artin's braid group B_∞ . Here we show how to use the results of Section 2 to prove the so-called Embedding Conjecture in many cases.

The group of left self-distributivity

Here we recall the definition of the group G_{LD} , and earlier results about this group and an associated monoid obtained using a combinatorial process called word reversing.

Definition. For every word w over \mathbf{A} , we define LD_w to be the partial operator on T_∞ that maps every sufficiently large term t to its LD-expansion $(t)w$. The monoid consisting of all operators LD_w equipped with reverse composition is denoted by \mathcal{G}_{LD}^+ .

The following equivalence results from the definition directly:

Lemma 5.1. [6] *Assume that t, t' are terms in T_∞ . Then the following are equivalent:*

- (i) *The term t' is an LD-expansion of the term t ;*
- (ii) *Some element of \mathcal{G}_{LD}^+ maps t to t' .*

The monoid \mathcal{G}_{LD}^+ , together with its symmetrized completion considered below, plays a significant role as it captures what can be called the intrinsic geometry of Identity (LD).

In addition to the operators LD_w , which correspond to expanding terms, we can also consider the inverse operators LD_w^{-1} which correspond to using (LD) in the direction $(xy)(xz) \mapsto x(yz)$. So, for every address α , we introduce LD_α^{-1} to be the inverse operator of LD_α (which is injective), and we consider the monoid \mathcal{G}_{LD} generated by all operators LD_α and LD_α^{-1} using reversed composition. By construction, every element in \mathcal{G}_{LD} is a finite product of operators LD_α and LD_α^{-1} . Using \mathbf{A}^{-1} for a disjoint copy of \mathbf{A} , where the copy of α is denoted α^{-1} , we can define $LD_{\alpha^{-1}}$ to be LD_α^{-1} , and then represent every element of \mathcal{G}_{LD} as LD_w , where w is a word over $\mathbf{A} \cup \mathbf{A}^{-1}$, *i.e.*, a finite sequence of signed addresses. We write $(\mathbf{A} \cup \mathbf{A}^{-1})^*$ for the set of all such words, of which $\phi \cdot 11^{-1} \cdot 0$ is a typical element. On the shape of Lemma 5.1, we have the following straightforward characterization:

Lemma 5.2. [6] *Assume that t, t' are terms in T_∞ . Then the following are equivalent:*

- (i) *The terms t and t' are LD -equivalent;*
- (ii) *Some element of \mathcal{G}_{LD} maps t to t' .*

The monoid \mathcal{G}_{LD} is not a group, because the product of LD_α and LD_α^{-1} is the identity mapping of its domain only, and not of T_∞ . This problem cannot be solved easily, as the product of two elements in \mathcal{G}_{LD} may be empty: so is for instance the operator LD_w for $w = \phi \cdot 1 \cdot \phi^{-1}$. However, the structure of \mathcal{G}_{LD} is fairly well known.

Proposition 5.3. [6] *Let \equiv^+ denote the congruence on \mathbf{A}^* generated by*

$$\begin{aligned}
\alpha \cdot \beta &\equiv^+ \beta \cdot \alpha && \text{for } \alpha \perp \beta && (\text{type } \perp) \\
\alpha 0 \beta \cdot \alpha &\equiv^+ \alpha \cdot \alpha 10 \beta \cdot \alpha 00 \beta && && (\text{type } 0) \\
\alpha 10 \beta \cdot \alpha &\equiv^+ \alpha \cdot \alpha 01 \beta && && (\text{type } 10) \\
\alpha 11 \beta \cdot \alpha &\equiv^+ \alpha \cdot \alpha 11 \beta && && (\text{type } 11) \\
\alpha 1 \cdot \alpha \cdot \alpha 1 \cdot \alpha 0 &\equiv^+ \alpha \cdot \alpha 1 \cdot \alpha, && && (\text{type } 1)
\end{aligned}$$

and \equiv be the congruence on $(\mathbf{A} \cup \mathbf{A}^{-1})^*$ generated by \equiv^+ together with $\alpha \cdot \alpha^{-1} \equiv \varepsilon$ and $\alpha^{-1} \cdot \alpha \equiv \varepsilon$ for every α .

- (i) *Assume that u, u' are words on \mathbf{A} . Then $u \equiv^+ u'$ implies $LD_u = LD_{u'}$.*
- (ii) *Assume that w and w' are words on $\mathbf{A} \cup \mathbf{A}^{-1}$ such that the domains of LD_w and $LD_{w'}$ are not disjoint. Then the following are equivalent:*
 - *There exists at least one term t satisfying $(t)w = (t)w'$;*
 - *For every term t , $(t)w$ and $(t)w'$ are equal when they exist;*
 - *The relation $w \equiv w'$ holds.*

Proposition 5.3 gives an optimal characterization of \mathcal{G}_{LD} , in particular in terms of presentation. In order to get rid of the limitations due to the operators LD_w being not everywhere defined, we introduce the group for which the above relations yield a presentation.

Definition. We denote by G_{LD} the group $(\mathbf{A} \cup \mathbf{A}^{-1})^* / \equiv$; the class of α in G_{LD} is denoted g_α . We denote by M_{LD} the monoid \mathbf{A}^* ; the class of α in M_{LD} is denoted g_α^+ .

Artin's braid group B_∞ is the quotient of G_{LD} obtained by collapsing all generators g_α such that α contains at least one 0, a property that explains the deep connection between braids and left self-distributivity.

When we consider LD-expansions only, *i.e.*, operators LD_w with w a word on \mathbf{A} , the domain of LD_w is never empty, and, if w, w' are words on \mathbf{A} , the domains of LD_w and $LD_{w'}$ are never disjoint [2]. So, we deduce from Proposition 5.3(ii):

Proposition 5.4. [6] *For all words u, u' on \mathbf{A} , $LD_{u'} = LD_u$ is equivalent to $u' \equiv u$, and, therefore, the monoid \mathcal{G}_{LD}^+ is isomorphic to the submonoid G_{LD}^+ of G_{LD} generated by the elements g_α .*

Now, we observe that the relations involved in the presentation of the group G_{LD} have a special syntactic form.

Definition. Assume that X is a nonempty set. We say that f is a *complement* on X if f is a function of $X \times X$ to the free monoid X^* generated by X satisfying $f(x, x) = \varepsilon$ for every x —where ε denotes the empty word. For f a complement on X , the monoid (*resp.* the group) with presentation $\langle X; \{xf(x, y) = yf(y, x); x, y \in X\} \rangle$ is said to be *associated* with f .

For f a complement on X , and w, w' words on $X \cup X^{-1}$, let us say that w is *f-reversible* to w' if w' can be obtained from w by repeatedly replacing some subwords of the form $x^{-1} \cdot y$ with the corresponding word $f(x, y)f(y, x)^{-1}$. For u, v words on X , we denote by $u \setminus v$ the (necessarily unique) word v' on X such that $u^{-1}v$ is *f-reversible* to $u'v'^{-1}$ for some word u' on X , if such a word exists. By construction, we have $x \setminus y = f(x, y)$ for all x, y in X , and the operation \setminus can be seen as an extension of f to arbitrary words on X .

Definition. Assume that f is a complement on X . We say that f is *convergent* if, for all words u, v on X , the word $u \setminus v$ exists; we say that f is *coherent* if, for all words u, v, w on X , we have

$$(u \setminus v) \setminus (u \setminus w) \setminus ((v \setminus u) \setminus (v \setminus w)) = \varepsilon.$$

Proposition 5.5. [8] *Assume that f is a convergent and coherent complement; let M and G be the associated monoid and the associated group, respectively.*

- (i) *The monoid M admits left cancellation.*
- (ii) *Let a, b be arbitrary elements of M , and u, v be words on X representing a and b respectively. Then a and b admit a right lcm in M , and the latter is represented by $u(u \setminus v)$.*
- (iii) *The monoid M embeds in the group G if and only if M admits right cancellation.*

Let us come back to the particular monoid M_{LD} . By definition, M_{LD} is associated with some complement henceforth denoted f on \mathbf{A} .

Lemma 5.6. [6] *The complement f is convergent and coherent.*

We deduce from Proposition 5.5(i), (ii):

Proposition 5.7. [6] *The monoid M_{LD} admits left cancellation, and right lower common multiples always exist in M_{LD} .*

The right lcm of two elements a, b of M_{LD} will be denoted by $a \vee b$. For u, v arbitrary words on \mathbf{A} , we shall write $u \vee v$ for the word $u(u \setminus v)$. This notation is coherent as, by Proposition 5.5(ii), if u, v represent a, b , then $u \vee v$ represents $a \vee b$.

We conclude this introductory section with a last object, namely, for each term t , a distinguished element denoted Δ_t of the monoid M_{LD} that describes the transformation of t into ∂t .

Definition. (i) For t a term, we let Δ_t be the element of M_{LD} inductively defined as follows: for t a variable, we have $\Delta_t = 1$; for $t = t_1 t_2$, we have $\Delta_t = a \text{sh}_1(\Delta_{s_1}) \text{sh}_0(\Delta_{s_2})$, where a is $1^{h-2} \cdot 1^{h-3} \cdots 1 \cdot \phi$ with h the length of the rightmost branch in t (viewed as a binary tree), $s_1 s_2$ is $(t)a$, and sh_α denotes the endomorphism of M_{LD} that maps g_γ^+ to $g_{\alpha\gamma}^+$ for every γ .

(ii) For $k \geq 0$, we define inductively $\Delta_t^{(k)}$ by $\Delta_t^{(0)} = 1$, and $\Delta_t^{(k)} = \Delta_t \Delta_{\partial t}^{(k-1)}$ for $k \geq 1$.

Proposition 5.8. [6] *For every term t , and every k , we have $\partial^k t = (t) \Delta_t^{(k)}$.*

The element $\Delta_t^{(k)}$ is an exact counterpart of Garside's braid Δ_n^k , with which it shares many algebraic properties. In particular, every element of M_{LD} is a left divisor of some element $\Delta_t^{(k)}$, as every positive braid is a left divisor of some braid Δ_n^k . However, braids are symmetric, and every positive braid is also a right divisor of some Δ_n^k , but the counterpart in M_{LD} is not true: some elements of M_{LD} are not right divisors of any element $\Delta_t^{(k)}$. We refer to [13] for various results about the elements $\Delta_t^{(k)}$, but such results are not needed for our present purpose.

The Embedding Conjecture

Proposition 5.4 leaves the description of the monoid \mathcal{G}_{LD}^+ incomplete, since it does not give an explicit presentation of the submonoid G_{LD}^+ of G_{LD} . On the other hand, \mathcal{G}_{LD}^+ is a quotient of the monoid M_{LD} by Proposition 5.3(i).

Conjecture 5.9. (Embedding Conjecture) *The monoid M_{LD} embeds in the group G_{LD} , i.e., for all positive words u, u' in \mathbf{A}^* , $u' \equiv u$ implies (and, therefore, is equivalent to) $u' \equiv^+ u$.*

We shall give several equivalent forms of the latter conjecture.

Proposition 5.10. *The following are equivalent:*

- (i) *The Embedding Conjecture is true;*
- (ii) *The monoid M_{LD} admits right cancellation;*
- (iii) *The monoid \mathcal{G}_{LD}^+ is isomorphic to the monoid M_{LD} , i.e., for all positive words u, u' in \mathbf{A}^* , $\text{LD}_{u'} = \text{LD}_u$ implies (and, therefore, is equivalent to) $u' \equiv^+ u$.*

Proof. That (i) implies (ii) is trivial. The converse implication follows from Proposition 5.5(iii) together with Lemma 5.6. That (i) implies (iii) directly follows from Proposition 5.4. Conversely, assume (iii). Let u, u' be words on \mathbf{A} satisfying $u' \equiv u$. By Proposition 5.3(i), we have $\text{LD}_u = \text{LD}_{u'}$, hence, by (iii), $u' \equiv^+ u$. ■

In order to study the latter statement more closely, we pose a definition. By Proposition 5.3(i), there is no ambiguity in defining, for a in M_{LD} , the operator LD_a to be value of LD_u where u is an arbitrary word on \mathbf{A} representing a .

Definition. Assume that a is an element of M_{LD} . We say that the Embedding Conjecture is true *for a* if the canonical projection of M_{LD} onto \mathcal{G}_{LD}^+ is injective on a , i.e., if $b \neq a$ in M_{LD} implies $\text{LD}_a \neq \text{LD}_b$.

Thus Conjecture 5.9 is the statement that the Embedding Conjecture is true for every element of a of M_{LD} .

Lemma 5.11. *Assume that the Embedding Conjecture is true for a .*

- (i) *For all a', b in M_{LD} , $ab = a'b$ implies $a' = a$.*
- (ii) *The Embedding Conjecture is true for every right divisor of a .*

Proof. (i) Assume that u represents a , and $uv \equiv^+ u'v$ holds. We deduce $uv \equiv u'v$, hence $u \equiv u'$, and $u \equiv^+ u'$ as the Embedding Conjecture is true for a .

(ii) Assume again that u represents a , and $u \equiv^+ u_1u_2$, and $u'_2 \equiv u_2$ hold. Then we have $u_1u'_2 \equiv u_1u_2$, hence $u_1u'_2 \equiv^+ u_1u_2$ as the Embedding Conjecture is true for a , and, finally, $u'_2 \equiv^+ u_2$ as M_{LD} is left cancellative. \blacksquare

The rest of this section is devoted to proving the Embedding Conjecture for some particular elements of M_{LD} . Let us first mention without proof that the Embedding Conjecture is true for every degree 1 element of M_{LD} , the latter being defined as those elements a such that the associated operator LD_a maps at least one term t to a degree 1 LD-expansion of t (and, in this case, it maps every term t to a degree 1 LD-expansion of that term). Such degree 1 elements in M_{LD} correspond to classical objects in braid theory, namely the divisors of Garside's universal elements Δ_n .

Confluent families

We shall now develop a new technique that enables us to deduce partial results on the Embedding Conjecture from the results of Section 2.

Definition. Assume that F is a family of terms. We say that F is *confluent* if, for all words u, v on \mathbf{A} , the family F contains the term $(t)u\upsilon v$ whenever it contains t , $(t)u$ and $(t)v$. We say that F is *locally confluent* if, for all addresses α, β , F contains $(t)\alpha\upsilon\beta$ whenever it contains t , $(t)\alpha$ and $(t)\beta$.

Considering confluent families is relevant for the Embedding Conjecture owing to the following criterion.

Lemma 5.12. *Let a be an element of M_{LD} . Then the following are equivalent:*

- (i) *The Embedding Conjecture is true for a ;*
- (ii) *For every term t in the domain of LD_a , the family of all LD-expansions of t of which $(t)a$ is an LD-expansion is confluent.*
- (iii) *There exist a term t in the domain of LD_a and a confluent family F containing t and $(t)a$ and such that $(t)a$ is terminal in F , i.e., no proper LD-expansion of $(t)a$ belongs to F .*

Proof. Assume that the Embedding Conjecture is true for a , and that t belongs to the domain of LD_a . Let F be the family of all LD-expansions of t of which $(t)a$ is an LD-expansion. Let w be a word on \mathbf{A} representing a . Assume $t', (t')u, (t')v \in F$. Then, by definition, there exist words w', u_1, v_1 on \mathbf{A} satisfying $t' = (t)w'$, $(t)a = (t')uv_1 = (t')vu_1$. Hence we have $LD_w = LD_{w'uv_1} = LD_{w'vu_1}$, and, therefore, $w \equiv w'uv_1 \equiv w'vu_1$. The Embedding Conjecture for a implies $w \equiv^+ w'uv_1 \equiv^+ w'vu_1$, hence $uv_1 \equiv^+ vu_1$ since M_{LD} admits left cancellation. By the properties of the operation \setminus , there exists a positive word w'' satisfying $v_1 = (u\setminus v)w''$ and $u_1 = (v\setminus u)w''$. Hence we have $(t)a = ((t')uvv)w''$, and, therefore, $(t')u\upsilon v \in F$. So F is confluent, and (i) implies (ii).

It is clear that (ii) implies (iii), as, by construction, the term $(t)a$ is terminal in every family satisfying the conditions of (ii).

Finally, assume that F is a confluent family, and $(t)a$ is terminal in F . Assume that u represents a , and $u' \equiv u$, hence $\text{LD}_{u'} = \text{LD}_u$, holds. By construction, we have $(t)u' = (t)u$. As $(t)u$ is an LD-expansion of $(t)u$ and $(t)u'$, the definition of a confluent family implies $(t)u \vee u' \in F$. The hypothesis that $(t)u$ is terminal in F implies $(t)u \vee u' = (t)u$, and the only possibility is $u \setminus u'$ to be the empty word, and so is $u' \setminus u$ by a symmetric argument. By the properties of \setminus , this implies $u' \equiv^+ u$, and the Embedding Conjecture is true for a . ■

By Proposition 5.5, lcm's in the monoid M_{LD} can be determined using the word reversing iterative technique. This results in a local characterization of confluent families, which involves letters instead of arbitrary words.

Lemma 5.13. *Assume that F is a family of terms. Then F is confluent if and only if it is locally confluent.*

Proof. By definition, (i) implies (ii). So, assume (ii). We establish using induction on m that F contains $(t)u \vee v$ whenever it contains $(t)u$ and $(t)v$ and $c(u, v) \leq m$ holds, where $c(u, v)$ is the number of elementary steps needed to reverse the word $u^{-1}v$. For $m = 0$, at least one of u, v is empty, and the result is obvious. Assume now $m \geq 1$. Then u and v are not empty. Let us write $u = \alpha \cdot u_0$, $v = \beta \cdot v_0$, with $\alpha, \beta \in \mathbf{A}$. There exist positive words $u_1, v_1, \dots, u_3, v_3$ such that $v_1 \cdot u_2^{-1}$ is obtained by word reversing from $u_0^{-1} \cdot (\alpha \setminus \beta)$, $v_2 \cdot u_1^{-1}$ is obtained by word reversing from $(\beta \setminus \alpha)^{-1} \cdot v_0$, and $v_3 \cdot u_3^{-1}$ is obtained by word reversing from $u_2^{-1} \cdot v_2$ (see Figure 5.1). By definition, the term $(t)\alpha \vee \beta$ belongs to F . Now, by construction, we have

$$c(u, v) = 1 + c(u_0, \alpha \setminus \beta) + c(v_0, \beta \setminus \alpha) + c(u_2, v_2),$$

hence each of the numbers $c(u_0, \alpha \setminus \beta)$, $c(v_0, \beta \setminus \alpha)$, and $c(u_2, v_2)$ is less than m . Using the induction hypothesis, we deduce successively that $(t)\alpha u_0 v_1$, $(t)\beta v_0 u_1$, and $(t)\alpha u_0 v_1 v_3$ belong to F . The latter term is $(t)u \vee v$. ■

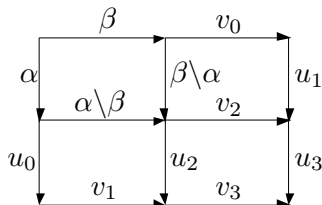


Figure 5.1. Induction for confluence

By gathering the previous results, we deduce the following criterion:

Proposition 5.14. *Assume that a is an element of M_{LD} . Then the following are equivalent:*

- (i) *The Embedding Conjecture is true for a ;*
- (ii) *There exists a locally confluent family of terms with a terminal term in the image of LD_a .*

This characterization establishes a connection between the Embedding Conjecture and the structure of LD-equivalence classes.

Proposition 5.15. *The Embedding Conjecture is equivalent to each of the following statements:*

- (i) *For all terms t, t' , and all addresses α, β , if t' is an LD-expansion of $(t)\alpha$ and $(t)\beta$, then it is an LD-expansion of $(t)\alpha \vee \beta$ as well.*
- (ii) *For all terms t, t' , and all words u, v on \mathbf{A} , if t' is an LD-expansion of $(t)u$ and $(t)v$, then it is an LD-expansion of $(t)u \vee v$ as well.*

Proof. Assume that the Embedding Conjecture is true, and assume that t' is an LD-expansion of $(t)u$ and $(t)v$. Then t' is an LD-expansion of t . By Lemma 5.12, the family F of all LD-expansions of t of which t' is an LD-expansion is confluent. By hypothesis, F contains $(t)u$ and $(t)v$, hence it contains $(t)uvv$, which means that t' is an LD-expansion of $(t)uvv$. So (ii) holds.

Next, (ii) implies (i) by definition.

Finally, assume (i), and let a be an arbitrary element of M_{LD} . Let t be a term in the domain of LD_a , and F be the family of all LD-expansions of t of which $(t)a$ is an LD-expansion. We claim that F is locally confluent. Indeed, assume that t' , $(t')\alpha$ and $(t')\beta$ belong to F . By construction, $(t')\alpha\vee\beta$ is an LD-expansion of t , and, as $(t)a$ is an LD-expansion both of $(t')\alpha$ and $(t')\beta$, then, by hypothesis, it must be an LD-expansion of $(t')\alpha\vee\beta$ as well. Hence $(t')\alpha\vee\beta$ belongs to F , which is locally confluent. We conclude that the Embedding Conjecture is true for a . ■

The problem for establishing the conditions of Proposition 5.15 is that no intrinsic characterization for the LD-expansions of a term is known in general. Now, in the case of perfect terms, such a characterization is given by Corollary 3.12.

Lemma 5.16. *Assume that t_0 is a perfect term. Then the family consisting of all terms included in t_0 is confluent.*

Proof. By Lemma 5.13, it suffices to prove that F is locally confluent. Assume that t , $(t)\alpha$ and $(t)\beta$ are included in t_0 . We consider the various possible cases. If the addresses α and β are orthogonal, then $\alpha\vee\beta$ is $\alpha\cdot\beta$, and the content of $(t)\alpha\cdot\beta$ is the union of the contents of $(t)\alpha$ and $(t)\beta$, so the result is clear.

Assume now that α and β are prefix-comparable, say α is a prefix of β for instance. The case $\beta = \alpha$ is trivial. If $\alpha 10$ or $\alpha 11$ is a prefix of β , the content of $(t)\alpha\cdot\beta$ is the union of the contents of $(t)\alpha$ and $(t)\beta$, and the result is clear again. There remain the cases $\beta = \alpha 1$ and $\alpha 0$ prefix of β . Assume first $\beta = \alpha 1$. We have $\alpha\setminus\beta = \alpha 1\cdot\alpha$. The content of the term $(t)\alpha\vee\beta$ is the union of the contents of $(t)\alpha$, $(t)\beta$, and of additional elements of the form abc where a appears in t at $\alpha 1101^*$, b appears at $\alpha 101^*$, and c appears below $\alpha 0$ (see Figure 5.2). Consider such an element. By construction, abc appears in an LD-expansion of t , hence we have $abc \sqsubset \bar{t} = \bar{t}_0$. Moreover, ab appears in $(t)\alpha$, bc appears in $(t)\beta$, hence both appear in t_0 , and Lemma 2.13 implies that abc appears in t_0 . So the result is true.

The final case is when $\alpha 0$ is a prefix of β , say $\beta = \alpha 0\gamma$. Then we have $\alpha\setminus\beta = \alpha 10\gamma\cdot\alpha 00\gamma$. The same argument as above shows that the content of $(t)\alpha\vee\beta$ is the union of the contents of $(t)\alpha$, $(t)\beta$, and of additional elements of the form abc where a appears in t at $\alpha 101^*$, b appears at $\alpha 0\gamma 101^*$, and c appears below $\alpha 0\gamma 0$. As in the case $\beta = \alpha 1$, we have $abc \sqsubset \bar{t}_0$, ab and bc appear in t , hence in t_0 , and Lemma 2.13 implies that abc appears in t_0 . ■

It follows from the analysis of the operators LD_w in [2] that, for every word w such that the operator LD_w is nonempty, there exists a pair of terms $(t_L(w), t_R(w))$, which is unique if we require in addition that the variables appear in increasing order when one looks at their leftmost occurrences enumerated from left to right, such that LD_w is the set of all substitutes of $(t_L(w), t_R(w))$, *i.e.*, the family of all pairs obtained from $(t_L(w), t_R(w))$ by replacing the variables by arbitrary terms. As positively equivalent words give the same operator, we can use the notation $t_L(a)$ and $t_R(a)$ for a in M_{LD} without ambiguity. Then we deduce the following criterion.

Proposition 5.17. *Assume that a is an element of M_{LD} . Then the following are equivalent:*

- (i) *The term $t_R(a)$ is full (hence perfect);*
- (ii) *The image of the operator LD_a contains at least one full term.*

If the above conditions are satisfied, then the Embedding Conjecture is true for a .

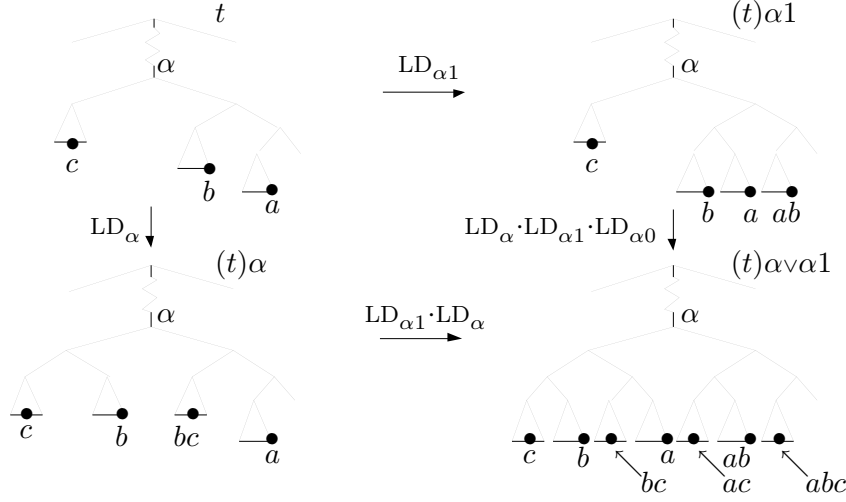


Figure 5.2. Confluent family

Proof. The equivalence of (i) and (ii) follows from Proposition 2.11, as the terms in the image of LD_a are the substitutes of $t_R(a)$. Assume that the term $t_R(a)$ is full. By construction, the term $t_L(a)$ is increasing, hence steep. Hence $t_R(a)$, which is LD-equivalent to $t_L(a)$, is steep as well, and, therefore, it is perfect. By Lemma 5.16, the family of all terms included in $t_R(a)$ is confluent, and, by construction, the term $t_R(a)$ is terminal in this family. By Proposition 5.14, we conclude that the Embedding Conjecture is true for a . ■

Using the results of Sections 2 and 3, we obtain the Embedding Conjecture for a large family of elements.

Proposition 5.18. *The Embedding Conjecture is true for every element of M_{LD} that is a right divisor of some element $\Delta_t^{(k)}$.*

Proof. The element $\Delta_t^{(k)}$ depends only on the skeleton of the term t , so we may assume without loss of generality that t is increasing. Then we have $(t)\Delta_t^{(k)} = \partial^k t$, a full term by Proposition 2.20. Hence $\Delta_t^{(k)}$ and, by Lemma 5.11, every right divisor of $\Delta_t^{(k)}$ in M_{LD} , is eligible for the criterion of Proposition 5.17. ■

As we observed that some elements of M_{LD} are not right divisors of any element $\Delta_t^{(k)}$, and, therefore, Proposition 5.18 is not yet a complete proof of the Embedding Conjecture.

Another application deals with those elements of M_{LD} that admit an expression involving generators g_α^+ with α of the form 1^i only. Such elements appear in connection with the natural section $s : \sigma_i \mapsto g_{1^{i-1}}^+$ of the canonical projection of M_{LD} onto the positive braid monoid B_∞^+ (this section s is not a homomorphism).

Definition. An element a of M_{LD} is said to be *braidlike* if it belongs to the submonoid generated by the elements $g_{1^i}^+$.

Lemma 5.19. *Assume that a is a braidlike element of M_{LD} . Then, for every perfect term t in the domain of LD_a , the term $(t)a$ is perfect. So, in particular, the term $t_R(a)$ is perfect.*

Proof. Assume that t is perfect, and $t' = (t)1^i$ holds. The term t' is steep since it is LD-equivalent to t , and the only problem is fullness. We look at those pairs (a', b') such that a' covers b' in t' and $\bar{t}' \sqsupset a'b'$, i.e., $\bar{t} \sqsupset a'b'$, holds. If a' and b' already appear in t , and a' covers b' in t , then the hypothesis that t is full implies that $a'b'$ appears in t , hence in t' .

Let a be the element that appears at $1^{i+1}01^*$ in t . By Proposition 1.12, three cases remain to be considered. The first case is $a' = a$ and b' appearing at $1^i0\beta$ in t . In this case, $a'b'$ appears in t' at $1^{i+1}0\beta$.

The second case is $a' = ab$, $b' = ac$ with b appearing at $1^i0\beta$ and c appearing at $1^i0\gamma$ and β covering γ . Assume $\bar{t} \sqsupset a'b'$, *i.e.*, $\bar{t} \sqsupset abc$. Then t being steep implies $\bar{t} \sqsupset bc$, and its being full implies that bc appears in t . If a covers bc in t , then $a(bc)$, which is $a'b'$, appears in t' as well, and we are done. Otherwise, bc appears in t below 1^i0 at some address say $1^i0\beta'$, and, in this case, $a'b'$ appears in t' at $1^{i+1}0\beta'$. The study of this case is therefore complete.

The last case is a' covering a in t . Now, by definition, this case is impossible, as the only addresses covering an address of the form 1^i01^* are those addresses of the form 1^{i+j} . ■

Proposition 5.20. *The Embedding Conjecture is true for every braidlike element of M_{LD} .*

The result follows from Proposition 5.17 directly by using Lemma 5.19.

References

- [1] S. BURCKEL, *The wellordering on positive braids*, J. Pure Appl. Algebra **120-1** (1997) 1–17.
- [2] P. DEHORNOY, *Free distributive groupoids*, J. P. Appl. Algebra **61** (1989) 123–146.
- [3] —, *Problème de mots dans les gerbes libres*, Theor. Comp. Sci. **94** (1992) 199–213.
- [4] —, *About the irreflexivity hypothesis for LD-magmas*, Logic Colloquium'90, J. Oikkonen & *al.* eds, Lect. Notes in Logic **2** (1993) 46–61.
- [5] —, *A canonical ordering for free LD systems*, Proc. Amer. Math. Soc. **122-1** (1994) 31–37.
- [6] —, *Braid groups and left distributive operations*, Trans. Amer. Math. Soc. **345-1** (1994) 115–151.
- [7] —, *A normal form for the free left distributive law*, Int. J. for Algebra & Computation **4-4** (1994) 499–528.
- [8] —, *Groups with a complemented presentation*, J. Pure Appl. Algebra **116** (1997) 115–137.
- [10] —, *From large cardinals to braids via distributive algebra*, J. Knot Theory & Ramifications **4-1** (1995) 33–79.
- [11] —, *On the syntactic algorithm for the word problem of left distributivity*, Algebra Univers. **37** (1997) 191–222.
- [12] —, *A fast method for comparing braids*, Advances in Math. **125** (1997) 200–235.
- [13] —, *The geometry monoid of left self-distributivity*, J. Pure Appl. Algebra, to appear.
- [14] P. DEHORNOY & L. PARIS, *Garside groups, a generalization of Artin groups*, Proc. London Math. Soc. **79-3** (1999) 569–604.
- [15] P. DELIGNE, *Les immeubles des groupes de tresses généralisés*, Invent. Math. **17** (1972) 273–302.
- [16] E. A. ELRIFAI & H. R. MORTON, *Algorithms for positive braids*, Quart. J. Math. Oxford **45-2** (1994) 479–497.
- [17] D. EPSTEIN & *al.*, *Word Processing in Groups*, Jones & Barlett Publ. (1992).
- [18] R. FENN, M.T. GREENE, D. ROLFSEN, C. ROURKE & B. WIEST, *Ordering the braid groups*, Pacific J. of Math., to appear.

- [19] F. A. GARSIDE, *The braid group and other groups*, Quart. J. Math. Oxford **20** No.78 (1969) 235–254.
- [20] D.M. LARUE, *On braid words and irreflexivity*, Algebra Univ. **31** (1994) 104–112.
- [21] R. LAVER, *The left distributive law and the freeness of an algebra of elementary embeddings*, Advances in Math. **91-2** (1992) 209–231.
- [22] —, *A division algorithm for the free left distributive algebra*, Oikkonen & al. eds, Logic Colloquium '90, Lect. Notes Logic **2** (1993) 155–162.
- [23] —, *On the algebra of elementary embeddings of a rank into itself*, Advances in Math. **110** (1995) 334–346.
- [24] —, *Braid group actions on left distributive structures and well orderings in the braid group*, J. Pure Appl. Algebra **108-1** (1996) 81–98.
- [25] C. ROURKE & B. WIEST, *Order automatic mapping class groups*, Pacific J. of Math., to appear.

Mathématiques, laboratoire SDAD, ESA 6081 CNRS
 Université Campus II, BP 5186, 14 032 Caen, France
 dehornoy@math.unicaen.fr
<http://www.math.unicaen.fr/~dehornoy/>