Towards a Higher-Dimensional String Theory for the Modeling of Computerized Systems

David Janin

LaBRI, CNRS UMR 5800
Laboratoire Bordelais de Recherche en Informatique

Rapport de recherche RR-1477-13

Towards a Higher-Dimensional String Theory for the Modeling of Computerized Systems

October 2013

David Janin,
LaBRI, IPB, Université de Bordeaux

Towards a Higher-Dimensional String Theory for the Modeling of Computerized Systems

David Janin*

LaBRI, Université de Bordeaux,
351, cours de la libération
F-33405 Talence, FRANCE
janin@labri.fr

Abstract. Recent modeling experiments conducted in computational music give evidence that a number of concepts, methods and tools belonging to inverse semigroup theory can be attuned towards the concrete modeling of time-sensitive interactive systems. Further theoretical developments show that some related notions of higher-dimensional strings can be used as a unifying theme across word or tree automata theory. In this invited paper, we will provide a guided tour of this emerging theory both as an abstract theory and with a view to concrete applications.

1 Introduction

As Mozart's character puts it in Milos Forman's masterpiece *Amadeus*, opera was for many years the only medium in which everyone could be talking at the same time and still manage to understand each other. Today, this has become common practice in communication networks.

Music theory has been developed empirically down the ages until it achieved status as a recognized discipline. It now provides us with the means for describing the underlying mechanisms and subtle interaction between musicians as they perform, based on complex combinatory rules relating to rhythm, harmony and melody. Computer science also aims to describe the subtle organization and treatment of data. It therefore follows that the study of modeling in the field of music might lead to the discovery of concepts, abstract tools and modeling principles which are applicable to modern computer engineering.

It is by developing language theoretical models of musical rhythmic structures that the author of this paper eventually re-discovered inverse semigroup theory: a part of semigroup theory that has been developed since the 50s. Further experiments, both in the field of computational music and in the field of formal language theory, provide evidence that such a theory can be further developed and attuned towards concrete computer engineering.

The purpose of this paper is to give an overview of these recent experiments and the underlying emerging ideas, methods and tools.

* CNRS temporary researcher fellow (2013-2014)

Mathematical frameworks *for* computer science and engineering.

In computer science and, more specifically, software or hardware engineering, *formal methods* are mathematically rigorous techniques for the specification, development and verification of computerized systems. Based on research fields in theoretical computer science as varied as type theory, automata and language theory, logic, these methods have already demonstrated their relevance for increasing the reliability of both software and hardware systems.

For instance, in functional programming, it is now common knowledge that proof and type theory does provide numerous metaphors and concepts that can be efficiently used by systems architects and developers. Typed functional languages such as *Haskell* or *OCAML* illustrate how deep mathematical theories can effectively be attuned towards relevant concepts which may be applied with ease. A similar observation can be made concerning data-base system design that now integrates highly usable methods and concepts that are deeply rooted in model theory.

Mathematical frameworks *for* interactive system design.

One of the most demanding application areas of automata and formal language theories is the domain of interactive system design. These systems are commonly viewed as state/transition systems, that is automata, and the behavior of these systems is commonly represented by the sets of their possible execution traces; that is, formal languages. A formal method such as *event B* [1], whose applicability to industry is clearly demonstrated regards to its use in automated public transport, is based on state/transition formalisms. With many features inherited from *method B* [3], it offers a particularly good example of how topics as varied as logic, proof theory, automata theory and formal languages can be *combined* and *shaped* towards applications [36].

However, while the mathematical frameworks that are available for functional programming or database design have been considerably and successfully developed and attuned towards applications over the last decades, the mathematics for interactive system modeling does not yet seem to have reached the same level of maturity. Since the early 80s, developing what became the *theory of concurrency*, many authors promoted the idea of modeling interactive systems by means of two distinct operators: the *sequential composition* and the *parallel composition* of system behaviors (see e.g. [11, 29]). However, examples of (distributed) system modeling show that such a distinction does not necessarily fit the abstraction/refinement methodology that system designers follow.

For instance, at the *abstract specification level* the well known distributed algorithm for leader election in a graph (see e.g. [4]) is based upon the *successive* execution of two global phases: the construction of a spanning tree followed by the pruning of that spaning tree. However, at the (distributed) *concrete execution level* these two phases overlap in time. Indeed, the pruning phase may start locally as soon as the spanning tree phase is locally completed. In other words, when composing two distributed algorithms one "after" the other, a composition of this type is neither purely parallel nor purely sequential: it is a sort of *mixed product* of some higher-dimensional models of spatiotemporal behaviors.

Towards a theory of higher dimensional strings.

This raises the question of whether there could be a mathematical theory of *higher-dimensional strings*. By describing quasi-crystals in physics [22], Kellendonk showed how such strings should be composed.

In one dimension, each string has two ends and, once a convention has been made on parity, a unique multiplication may be defined, called string concatenation, which enables pairs of strings to be multiplied, thereby giving rise to the structure of a free monoid. In higher dimensions, there is no uniquely defined way of composing patterns. Kellendonk hits upon the idea of labeling patterns by selecting two tiles of the pattern: one to play the role of the input and the other that of output. By using this labeling, a unique composition can be defined. The resulting semigroup is no longer free but is some kind of inverse semigroup [22, 23].

In music system programming where the time vs space problematic also appears (see e.g. [14]), similar ideas, also rooted in inverse semigroup theory, have led to the definition of a higher abstraction layer into which both strings and streams may be embedded into a single object type: tiled streams [20]. As a result, this new programming layer allows for the combination of both (sequential) strings and (parallel) streams with a single mixed product: the tiled product.

By capitalizing on theoretical developments within the general theory of inverse semigroups (see [25] for instance) and the associated emerging notion of higher-dimensional strings [22, 23], we thus aim at developing the idea of an inverse semigroup theory *for* computer science much in the same way that the development of logic *for* computer science is advocated in [36].

In this paper, we provide a guided tour of the first experiments conducted in this direction, both as an abstract theory and with a view to concrete applications.

2 From partial observations to inverse semigroups

On the footpath of Kellendonk for describing the local structures of quasi-crystals that leads to the discovery of (a notion of) tiling semigroups [22, 23], we aim here at illustrating how inverse semigroup structures naturally arise by performing *partial observations* of (the local structure of) computerized *system behavior spaces*.

Partial observations.

We assume that the behavior of a system can be viewed as the traversal of a complex space $\mathcal{S}$ of all potential behaviors that describe both the way data are structured in space but also the way they can evolve as time passes. Then, a *partial observation* is defined by the observation resulting of an out of time[1] traversal of such a space $\mathcal{S}$, that induces some *domain* of the traversal, together with the starting point of such a traversal, called the *input root*, and the end

[1] time flows does not matter at that stage

point of such a traversal, called the *output root*. Such a (meta) notion of partial observation may be depicted as in Figure 1. Of course, the exact nature of a

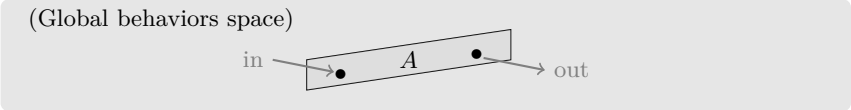

Fig. 1. A partial observation A

partial observation, that is, the nature of its domain and the associated input and input roots, depends on the mathematical nature of the space $\mathcal{S}$.

For instance, in the approach of Kellendonk [22], the space $\mathcal{S}$ is a quasi-crystal, that is a collection of distinct tiles that tiles the Euclidian space $\mathbb{R}^d$. Then, a partial observation is defined as a set T of tiles of $\mathcal{S}$ that cover a connected subset of $\mathbb{R}^d$ together with two distinguished tiles $I \in T$ and $O \in T$ acting as input and output roots (see [22] and [23] for more details).

In free inverse monoids [32, 30], the space $\mathcal{S}$ is defined as the Cayley graph of the free group $FG(A)$ generated by a given alphabet A. Then, partial observations are defined as finite connected subgraphs of that Cayley graph with two distinguished vertices: one for the input root and the other for the output root.

In this case, partial observations are *birooted trees*, that is, finite directed tree-shaped graphs with deterministic and co-deterministic A-labeled edges. Extension with labeled vertices are considered in [18] in connection with tree walking automata and also in [16] in connection with non deterministic tree automata.

Restricting to linear birooted trees, we obtain what may be called *birooted words*, structures that already appear in the 70's as elements of the monoid of McAlister [26].

Observation composition.

Two observations A and B can be composed in a two step procedure defined by, first, a *synchronization* that amounts to sewing the output root of A with the input root of B, followed secondly by a *fusion* that amounts to merging the sub-domains that may overlap. Such a (meta) notion of composition of observations can be depicted as in Figure 2.

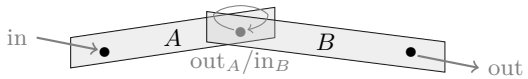

Fig. 2. The product $A \cdot B$ of two partial observations A and B

In many settings, the fusion operation may fail hence, apriori, the product of two observations is a partial product. In such a case, it is completed by adding an

undefined observation, denoted by 0, that simply acts as zero, i.e. $A \cdot 0 = 0 = 0 \cdot A$ for every observation A. It is an easy exercise to check that such a zero element is necessarily unique and idempotent, i.e. $0 \cdot 0 = 0$.

It may also be the case that the structure of observations allows for the definition of a special element, the unit, denoted by 1, that is neutral for the product, i.e. $A \cdot 1 = A = 1 \cdot A$ for every observation A. It is an easy exercise to check that, necessarily, such a neutral element is also unique and idempotent.

The semigroup/monoid of partial observations.

In general, nothing guarantees that such a product is associative so we simply assume this, that is, we assume that $(A \cdot B) \cdot C = A \cdot (B \cdot C)$ for every observation A, B and C. Though simple, let us insists on the fact that associativity is the first key property for a robust and usable implementation for it means that a complex product can be specified and computed in any order.

In algebra, the resulting structure is known as a *semigroup*. In the case there is also a unit, it is a *monoid*.

Without any further assumptions, two special observations can already be defined out of a given one. Indeed, given an observation A, we may also consider the observation A^R, called the *reset* or *right projection* of A, defined by moving the output root to the input root, and, symmetrically, the observation A^L, called the *co-reset* or *left projection* of A, defined by moving the input to the output root. The resulting observations are depicted in Figure 3. We observe that the

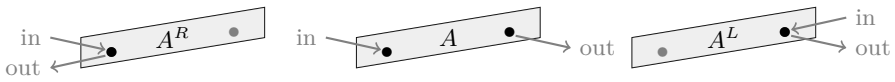

Fig. 3. The reset and co-reset operators

mapping $A \mapsto A^L$ and $A \mapsto A^R$ are indeed projection since, for every observation A, we have $(A^L)^L = A^L = (A^L)^R$ and $(A^R)^L = A^R = (A^R)^R$. The left and right projection are extended to zero by taking $0^L = 0 = 0^R$. It is then possible to define two derived operators, called *fork* and *join*, by $fork(A, B) = A^R \cdot B$ and $join(A, B) = A \cdot B^L$. These derived operators are depicted in Figure 4.

Fig. 4. The $fork$ and $join$ derived operators

Interpreting for a while the paths from input to output roots as time flows, in the $fork$ operation, everything looks as if the two observations "start" at the

same "time" in their input roots. Similarly, in the *join* operation, it looks as if the two observations "end" at the same "time" in their output roots.

More generally, at the *abstract syntactic level*, the composition $A \cdot B$ of two successive observations A and B, actually admits, at the *concrete model level*, some parallel flavored characteristics induced by the possible overlaps that may arise in the resulting structure. In other words, everything looks as if the observation product we are defining here at some meta level is a good candidate for the *mixed* sequential and parallel operator we are looking for as mentioned above.

Example. In computational music, such an approach is already used as follows. Every finite audio stream is enriched with input and output synchronization marks that, when mixing two streams, allows for automatically positioning them in time, one with respect to the other. Doing so, synchronization specifications have thus been internalized in some sense into the audio streams themselves, much like music bars in a music score. The resulting algebra [2], with product, left and right projections, and some other operators acting on tempos, provides a fairly robust and versatile language for music composition. Further implementation experiments have been conducted in [21].

Product by synchronized superposition.

In the concrete inverse semigroups mentioned above, the definition of the product of two partial observations can be refined as follows:

Definition 1. The product $A \cdot B$ of two observations A and B is defined to be the observation C (assumed to be uniquely defined up to isomorphism when it exists) such that there exist two embeddings $\varphi_A : A \to C$ and $\varphi_B : B \to C$ so that:

- the input root of A (resp. the output root of B) is mapped to the input root (resp. the output root) of C,
- the output root of A and the input root of B are mapped to the same image,
- the domain of C is the union of the domains of $\varphi_A(A)$ and $\varphi_B(B)$,

the product being completed by zero when no such observation C exists.

Remark. In other words, in that case, everything looks as if the product $A \cdot B$ is performed by translating the two observations A and B in such a way that their output and input root coincide. Then, the fusion just amounts to checking that the resulting overlapping subdomains are isomorphic.

More generally, Stephen's representation theorem of inverse semigroups [34] tells that every elements of every inverse semigroup have a *graphical representations*. In view of applications, this fact that is of high interest. However, it may be the case that the product induces a graph composition that does not fit the above definition.

In other words, the informal point of view that has been previously given is still worthy of being kept in mind when looking for new instances of models of partial observations and the related compositions.

Resulting idempotents (system states).

When the observation product is defined by synchronized superposition, as well as with Stephen graphical representation of inverse semigroups, and probably in many other settings still to be defined and studied, elements for which the input and output root coincide plays an especially important rôle.

In a context even more general than inverse semigroups as shown in the next section, these elements are be called *subunits*. By definition, both the left and right projection A^L and A^R of an arbitrary observation A, are subunits.

The fact is that subunits are both *idempotent* and *commute*, that is, for every elements E and F with identical input and output roots, we have $E \cdot E = E$ and $E \cdot F = F \cdot E$. They can thus be partially ordered as follows. For every two subunit observation E and F, we say that E is smaller than F, which is written $E \leq F$, when $E = E \cdot F$.

One can easily check that this indeed defines a partial order relation, that is, a reflexive, transitive and antisymmetric relation. Moreover, the meet $E \wedge F$ of every two subunits E and F, that is the greatest subunit below both E and F, exists and can simply be computed by $E \wedge F = E \cdot F$. Such an order relation is called the *natural order* for it can be defined within the semigroup itself [31].

Example. Potential application oriented views of such an order relation are numerous. For instance, one may view every subunit as the partial description of the *state* of the modeled system. The underlying domain of a subunit E tels how far in the past, in the future and also in the present time the system has been observed in E.

Then, the natural order tells us how wide such an observation is: the wider the observation is, the lower it is in the natural order, until it becomes incoherent. Indeed, the lowest subunit is zero. Then, the product of two partial states/subunit observations simply describes the union of the descriptions associated with these two partial states.

Quite strikingly, as recently observed by Dominique Méry and the author, when observations are viewed as models of behaviors, realizing a system that goes from a state specification S_0 to a state specification S_1 amount to finding a system behavior X such that $0 < (S_0 \cdot X)^L \leq S_1$. Some modeling experiments of similar ideas have been conducted in [6].

The inverse semigroup of partial observations.

The inverse semigroup of observations arises when we assume, as is the case with the product by superposition, that, for every observation A, the observation A^{-1} obtained just by inverting the input and the output root of the observation A, as depicted in Figure 5, satisfies the following properties:

$$A \cdot A^{-1} \cdot A = A \text{ and } A^{-1} \cdot A \cdot A^{-1} = A^{-1} \tag{1}$$

for every observation A, and,

$$A \cdot A^{-1} \cdot B \cdot B^{-1} = B \cdot B^{-1} \cdot A \cdot A^{-1} \tag{2}$$

for every observation A and B.

In semigroup theoretical terms, Property (1) says that elements A and A^{-1} are *semigroup inverses* one with respect to the other. Property (2) says that the resulting idempotent elements of the form $A \cdot A^{-1}$ commute. Indeed, by

Fig. 5. An observation and its inverse

associativity, we have $(A \cdot A^{-1}) \cdot (A \cdot A^{-1}) = (A \cdot A^{-1} \cdot A) \cdot A^{-1}$ hence, by applying Property (1), we have $(A \cdot A^{-1}) \cdot (A \cdot A^{-1}) = A \cdot A^{-1}$. Then we also have $A^R = A \cdot A^{-1}$ and $A^L = A^{-1} \cdot A$, as depicted in Figure 6. Even more, we have $A^R \cdot A = A = A \cdot A^L$, i.e. these projections are (the least) local units of the observation A. Forming an inverse semigroup, the *natural order* on observations

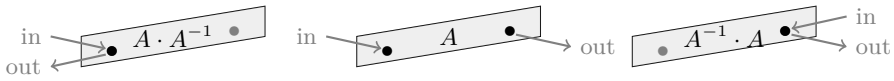

Fig. 6. Inverses and left and right projections

is extended to all semigroup elements by letting $A \leq B$ when $A = A^R \cdot B$ or, equivalently, $A = B \cdot A^L$. This order relation is depicted in Figure 7. One can easily check that the natural order is stable under product, i.e. if $A \leq B$ then $A \cdot C \leq B \cdot C$ and $C \cdot A \leq C \cdot B$ for every observations A, B and C. In the concrete cases mentioned so far, we even have $A \leq B$ when there exists an embedding $\varphi : B \to A$ that preserves input and output roots. Let us mentioned that many

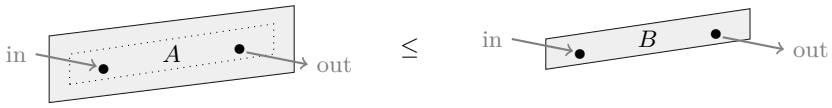

Fig. 7. The natural order

other properties of inverse semigroups can be found in [25] that provide a much deeper (and precise) description of such a peculiarly rich theory.

Remark. It may be the case that the partial observations described so far are defined on a structure space that forces input root and output root to be ordered, the input root always preceding or being equal to the output roots, as this can be the case with positive birooted words studied in [17] or positive birooted trees that are known to be the elements of the free ample monoids [10].

The resulting semigroups are no longer inverse semigroups for inverses themselves are not elements of these semigroups. However, these semigroups are *quasi-inverse* in some sense: the natural order and the left and right projections are still available.

It must mentioned that, since the late 70s in semigroups theory, Fountain et al. have develop a theory of semigroups with local units (see [9, 24, 12, 5]), quasi-inverse in the above sense, which underlying concepts are of high interest for developing a language theory of higher-dimensional strings as in the next section.

3 On languages of higher-dimensional strings

In view of applications to computer science, there is a need for a language theory for observations. Indeed, when specifying the expected behavior of a system one generally describes some characteristic properties of its correct behavior. In general, especially in the context of an abstraction/refinement designed method, there is no reason for such specifications to characterize a single possible behavior. In this section, we thus aim at defining adequate algebraic tools for a language theory of observations.

Recognizability and logic: the classical approaches.

In the absence of concrete structures, as it is the case in the general and abstract setting described here, algebraic recognizability is a major tool for defining languages, that is, subsets of peculiar semigroups or monoids.

More precisely, a mapping $\varphi : S \to T$ from a semigroup S to a semigroup T is a semigroup morphism when $\varphi(x \cdot y) = \varphi(x) \cdot \varphi(y)$ for every x and $y \in S$. When both S and T are monoid, the semigroup morphism φ is a monoid morphism when $\varphi(1) = 1$. A subset $L \subseteq S$ of a semigroup S (resp. monoid S) is said to be recognizable by a semigroup (resp. monoid) T when there exists a semigroup (resp. monoid) morphism such that $X = \varphi^{-1}(\varphi(X))$.

In the case of languages of words, that is subsets of the free monoid, the algebraic notion of recognizability does coincide with the notion of recognizability by finite state automata.

Indeed, given a morphism $\varphi : A^* \to S$ with finite monoid S, one can define the automaton $\mathcal{A}_S$ with set of state S, initial state 1, and deterministic transition $s \xrightarrow{a} t$ whenever $s \cdot \varphi(a) = t$. Then, for every $X \subseteq S$, the language $\varphi^{-1}(X)$ exactly correspond to the language of words recognized by the automaton $\mathcal{A}_S$ with accepting states X. Incidentally, this observation also gives a fairly efficient way to compute $\varphi(w)$ for every $w \in A^*$ given as input.

However, if we restrict target semigroups to be inverse as in [27], or if we consider languages of free inverse monoids as in [33], besides the inherent interest of the results established in these studies, the notion of recognizability by or even from inverse semigroups collapses. Indeed, when logical definability in monadic second order (MSO) logic [35] is available, as with birooted words [17] or birooted trees [16], the languages that are recognizable by means of finite monoids and

morphisms are far simpler than the languages definable by means of an MSO formula (see [17, 33, 16]).

This comes from the fact that the direct image of an inverse semigroup by a semigroup morphism is an inverse semigroup hence the automaton $\mathcal{A}_S$ defined above is of a very special kind: it is reversible in some sense.

Relaxing morphisms into premorphisms.

A remedy to the above mentioned collapse is based on the fact that inverse monoids are partially ordered by means of their natural order. Then, we can relax the morphism condition into the following condition. A mapping $\varphi : S \to T$ from one partially ordered monoid S in a partially ordered monoid T is a *premorphism* (or $\vee$-premorphism in [13]) when $\varphi(x \cdot y) \leq \varphi(x) \cdot \varphi(y)$ for every x and $y \in S$ with $\varphi(1) = 1$.

Then, as with morphisms, it is tempting to define recognizability by partially ordered monoid via premorphism $\varphi : S \to T$. However, premorphism condition alone is too weak. It can be shown that there are premorphisms from finitely generated inverse monoids into finite monoids that are even not computable [15]. The difficulty thus lies in defining an adequate restriction of both premorphisms and partially ordered monoids that induces a notion of recognizability that is both sufficiently expressive and still computable.

Our proposal is based on the fact that, in a finitely generated inverse monoid such as monoids of birooted words [19] or monoids of birooted trees [16], every element can be defined out from the finite generators, a notion of disjoint product and left and right projections. In other words, neither arbitrary product nor inverses themselves are needed to generate these monoids.

Remark. Though premorphisms have been known and used for quite some time in inverse semigroup theory [28], it seems that their use for language recognizability has been first proposed by the author himself in [15].

Adequate monoids and premorphism.

The recognizers we use instead of semigroups (or monoids) are adequately ordered monoids. Following the research track initiated by Fountain, our proposal is based on ordered monoids that are quasi-inverse in the sense that, though without inverses themselves, these monoids are still equipped with left and right projection that behaves *like* xx^{-1} and $x^{-1}x$.

Definition 2. A *partially ordered* monoid is a monoid S equipped with a stable partial order relation $\leq$, that is, for every x, y and $z \in S$, if $x \leq y$ then $xz \leq yz$ and $zx \leq zy$. Then, an *adequately ordered monoid* is a partially ordered monoid so that, for every element $x \in S$:

- if $x \leq 1$ then $xx = x$, i.e. subunits are idempotent,
- both $x^L = \min\{z \leq 1 : xz = x\}$ and $x^R = \min\{z \leq 1 : zx = x\}$ exist.

It is an easy exercice to check that, by stability, subunits also commute and, ordered by the order relation, form a meet semi-lattice with product as meet. When S is finite, subunits even form a complete lattice which suffices to guarantee the existence of left and the right projections.

Examples. Every monoid S, trivially ordered with a new leaset element zero, is an adequately ordered monoid with projections $x^L = 1 = x^R$ for every $x \in S$. Also, as already observed, every inverse semigroup S is adequately ordered by the natural order with $x^L = x^{-1}x$ and $x^R = xx^{-1}$ for every $x \in S$. The monoid $\mathcal{P}(Q \times Q)$ of relations over a set Q and ordered by relation inclusion is also adequately ordered.

The notion of observation disjoint product, though defined in a quite adhoc way in each concrete settings, essentially amounts to saying that the product $A{\cdot}B$ of two observations A and B is disjoint when it is non zero and the intersection of (the embedding of) their domains in the resulting product is limited to the output root of A that has been sewn with the input root of B. An example of a disjoint product of this type is depicted in Figure 8.

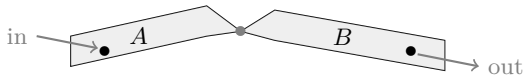

Fig. 8. Disjoint product of two observations

Definition 3. A adequate premorphism is a premorphism $\varphi : S \to T$ from a given concrete monoid S of observations into an adequately ordered monoid T such that:

- if $A \leq B$ then $\varphi(A) \leq \varphi(B)$,
- $\varphi(A^L) = (\varphi(A))^L$ and $\varphi(A^R) = (\varphi(A))^R$,
- if the product $A \cdot B$ is disjoint then $\varphi(A \cdot B) = \varphi(A) \cdot \varphi(B)$.

for every observations A and $B \in S$.

Remark. An immediate consequence of such a definition is that in the case observations can be generated from a finite set of elementary observations, disjoint products and left and right projections, i.e. every observations admits a good representations, then the image $\varphi(A)$ by any adequate premorphism φ into a finite adequately ordered monoid is computable in linear time in the size of the good representations. This happens with birooted words or trees [15, 16].

Quasi-recognizability.

We are now ready to define quasi-recognizability, that is, recognizability by adequate premorphisms and ordered monoids.

Definition 4. A language $L \subseteq S - 0$ is *quasi-recognizable* when there is an adequate premorphism $\varphi : S \to T$ from S to a finite adequately ordered monoid T such that $L = \varphi^{-1}(\varphi(L))$.

Such an extension of language recognizability has been proposed in [17] and studied further in [19] and [16]. It happens that it is quite robust in the sense that, in the studied case of birooted words and birooted trees where definability in MSO is available, we prove that:

Theorem 5 (see [19, 16]). *When S is the concrete monoid of birooted words or (labeled) birooted trees, a language $L \subseteq S$ is quasi-recognizable if and only if it is a finite boolean combination of upward closed MSO definable languages.*

Proving these results is achieved via a fairly simple extension of the notion of finite state non deterministic automata to birooted structures. These finite state automata are shown to characterize MSO definable upward closed languages or, equivalently, upward closed quasi-recognizable languages.

Generalized to languages, finding a set X of possible behaviors from a set of initial state S_0 to a set of possible final state S_1 amounts still to solving an inequality of the form $0 \subset (S_1 \cdot X)^L \leq S_1$ with product and projections extended in a point wise manner.

Remark. It must be mentioned that walking automata on trees also induce premorphisms that must satisfy certain kinds of (greatest fixpoint) equations that, in turn, make the premorphism (simply) computable [18].

Quite interestingly, these premorphisms do not preserve disjoint products but, instead, some extension of the well known notion of restricted products in inverse semigroup theory. Since the theory of recognizability by premorphisms is still in its infancy, it may be the case that this fixpoint approach could well be much more fruitful than the one presented here.

4 Conclusion

It shall be clear that the higher-dimensional string theory proposed here is still in its early stages of development. Further studies are currently conducted towards deeper modeling experiments, extension to infinite structures [7] and towards developing the underlying algebraic tools [8].

For instance, our logical characterization stated above ensures that the quasi-recognizable languages of positive birooted words or trees are closed under product or iterated product. Yet, providing direct algebraic or automata theoretic proof of these facts has been surprisingly difficult [8]. This means that the algebraic setting that is proposed here needs to be understood in a much greater depth.

More generally, every mathematical framework that can be used in computer science can be seen as a spotlight that helps us understanding the nature of the objects computer science and engineering may handle. The various experiments and theoretical developments that have been sketched in this paper tend to prove that inverse semigroup theory may provide, in the long term, an especially bright light for such a purpose.

Acknowledgment

Developing a trans-disciplinary research program that also aims at achieving advances in each of the specific research fields that are covered would probably be simply impossible without the support, encouragement and knowledge of true

experts in the covered fields. The author wishes to express his deepest gratitude to all those he has had the pleasure of meeting and discussing questions with during the last two years.

More specifically, to name but a few, grateful thanks to Myriam DeSainte-Catherine, Florent Berthaut, Jean-Louis Giavitto and Yann Orlarey in the field of Computational Music, Marc Zeitoun, Victoria Gould, Mark Lawson and Sylvain Lombardy in the field of Semigroup Theory, Anne Dicky and Dominique Méry in the field of Formal methods, Sylvain Salvati and Paul Hudak in the field of Typed Functional Programming, and, last, Lucy Edwards for her invaluable knowledge of the English language itself.

The pleasure and honor the author felt when being kindly invited by Professor Viliam Geffert to give a lecture at SOFSEM 2014 in the beautiful landscape of the High Tatras in Winter is to be shared with all of them.

References

1. Abrial, J.R.: Modeling in Event-B - System and Software Engineering. Cambridge University Press (2010)
2. Berthaut, F., Janin, D., Martin, B.: Advanced synchronization of audio or symbolic musical patterns: an algebraic approach. International Journal of Semantic Computing **6**(4) (2012) 409–427
3. Cansell, D., Méry, D.: Foundations of the B method. Computers and Informatics **22** (2003)
4. Chalopin, J., Métivier, Y.: An efficient message passing election algorithm based on mazurkiewicz's algorithm. Fundam. Inform. **80**(1-3) (2007) 221–246
5. Cornock, C., Gould, V.: Proper two-sided restriction semigroups and partial actions. Journal of Pure and Applied Algebra **216** (2012) 935–949
6. Dicky, A., Janin, D.: Modélisation algébrique du diner des philosophes. In: Modélisation des Systèmes Réactifs (MSR), in Journal Européen des Systèmes Automatisés (JESA Volume 47 - no 1-2-3/2013). (november 2013)
7. Dicky, A., Janin, D.: Embedding finite and infinite words into overlapping tiles. Technical report, LaBRI, Université de Bordeaux (October 2013)
8. Dubourg, E., Janin, D.: Algebraic tools for the overlapping tile product. Technical report, LaBRI, Université de Bordeaux (October 2013)
9. Fountain, J.: Right PP monoids with central idempotents. Semigroup Forum **13** (1977) 229–237
10. Fountain, J., Gomes, G., Gould, V.: The free ample monoid. Int. Jour. of Algebra and Computation **19** (2009) 527–554
11. Hoare, C.: Communicating Sequential Processing. Prentice-Hall International Series in Computer Science. Prentice-Hall International (1985)
12. Hollings, C.D.: From right PP monoids to restriction semigroups: a survey. European Journal of Pure and Applied Mathematics **2**(1) (2009) 21–57
13. Hollings, C.D.: The Ehresmann-Schein-Nambooripad Theorem and its successors. European Journal of Pure and Applied Mathematics **5**(4) (2012) 414–450
14. Hudak, P.: A sound and complete axiomatization of polymorphic temporal media. Technical Report RR-1259, Department of Computer Science, Yale University (2008)

15. Janin, D.: Quasi-recognizable vs MSO definable languages of one-dimensional overlapping tiles. In: Mathematical Found. of Comp. Science (MFCS). Volume 7464 of LNCS. (2012) 516–528
16. Janin, D.: Algebras, automata and logic for languages of labeled birooted trees. In: Int. Col. on Aut., Lang. and Programming (ICALP). Volume 7966 of LNCS., Springer (2013) 318–329
17. Janin, D.: On languages of one-dimensional overlapping tiles. In: Int. Conf. on Current Thrends in Theo. and Prac. of Comp. Science (SOFSEM). Volume 7741 of LNCS. (2013) 244–256
18. Janin, D.: Overlaping tile automata. In: 8th International Computer Science Symposium in Russia (CSR). Volume 7913 of LNCS., Springer (2013) 431–443
19. Janin, D.: Walking automata in the free inverse monoid. Technical Report RR-1464-12, LaBRI, Université de Bordeaux (2013)
20. Janin, D., Berthaut, F., DeSainte-Catherine, M., Orlarey, Y., Salvati, S.: The T-calculus : towards a structured programming of (musical) time and space. In: Workshop on Functional Art, Music, Modeling and Design (FARM), ACM Press (2013)
21. Janin, D., Berthaut, F., DeSainteCatherine, M.: Multi-scale design of interactive music systems : the libTuiles experiment. In: Sound and Music Computing (SMC). (2013)
22. Kellendonk, J.: The local structure of tilings and their integer group of coinvariants. Comm. Math. Phys. **187** (1997) 115–157
23. Kellendonk, J., Lawson, M.V.: Tiling semigroups. Journal of Algebra **224**(1) (2000) 140 – 150
24. Lawson, M.V.: Semigroups and ordered categories. I. the reduced case. Journal of Algebra **141**(2) (1991) 422 – 462
25. Lawson, M.V.: Inverse Semigroups : The theory of partial symmetries. World Scientific (1998)
26. Lawson, M.V.: McAlister semigroups. Journal of Algebra **202**(1) (1998) 276 – 294
27. Margolis, S.W., Pin, J.E.: Languages and inverse semigroups. In: Int. Col. on Aut., Lang. and Programming (ICALP). Volume 172 of LNCS., Springer (1984) 337–346
28. McAlister, D., Reilly, N.R.: E-unitary covers for inverse semigroups. Pacific Journal of Mathematics **68** (1977) 178–206
29. Milner, R.: Communication and concurrency. Prentice-Hall (1989)
30. Munn, W.D.: Free inverse semigroups. Proceeedings of the London Mathematical Society **29**(3) (1974) 385–404
31. Nambooripad, K.S.S.: The natural partial order on a regular semigroup. Proc. Edinburgh Math. Soc. **23** (1980) 249Ű260
32. Scheiblich, H.E.: Free inverse semigroups. Semigroup Forum **4** (1972) 351–359
33. Silva, P.V.: On free inverse monoid languages. ITA **30**(4) (1996) 349–378
34. Stephen, J.: Presentations of inverse monoids. Journal of Pure and Applied Algebra **63** (1990) 81–112
35. Thomas, W.: Chap. 7. Languages, automata, and logic. In: Handbook of Formal Languages, Vol. III. Springer-Verlag, Berlin Heidelberg (1997) 389–455
36. Thomas, W.: Logic for computer science: The engineering challenge. In Wilhelm, R., ed.: Informatics - 10 Years Back, 10 Years Ahead. Volume 2000 of LNCS., Springer (2001) 257–267