Fixed point characterization of infinite behavior of finite state systems

Damian Niwiński[1]
Institute of Informatics, University of Warsaw
niwinski@mimuw.edu.pl

November 21, 1996

[1]Supported by Polish KBN grant No. 2 P301 009 06.

Abstract

Infinite behavior of nondeterministic finite state automata running over infinite trees or more generally over elements of an arbitrary algebraic structure is characterized by a calculus of *fixed point terms* interpreted in powerset algebras. These terms involve the least and greatest fixed point operators and disjunction as the only logical operation. A tight correspondence is established between a hierarchy of Rabin indices of automata and a hierarchy induced by alternation of the least and greatest fixed point operators. It is shown that, in the powerset algebra of trees constructed from a set of functional symbols, the fixed point hierarchy is infinite unless all the symbols are unary (i.e. trees are words). It is also shown that an interpretation of a closed fixed point term in any powerset algebra can be factorized through the interpretation of this term in the powerset algebra of trees, from which it is deduced that the question whether a term denotes always $\emptyset$ can be answered in polynomial time.

Contents

Introduction

Nonterminating behavior of a finite state system is a mathematical abstraction of a number of phenomena that occur in information processing, such as continuously operating concurrent programs, network communication protocols or nonstabilizing asynchronous digital circuits.

Formalisms that have been proposed for specifying and reasoning about nonterminating behavior of programs can be roughly divided into three groups : various kinds of automata, temporal and modal logics, and fixed point calculi, i.e. formalisms based on explicit notation for inductive and co–inductive definitions. All of these are closely related (see [49, 11] for survey articles). Logics provide a nice tool for specification as they are close to human thinking; on the other hand, the relative algorithmic issues, as e. g. the problems of satisfiability and model checking, present some conceptual difficulties that are usually solved by reducing these questions to the emptiness problems for suitable automata. Automata are usually better tractable algorithmically due to a straightforward rather than inductive semantics; on the other hand, the automata notation does not always reflect the structural complexity of specified properties; therefore the use of automata as a specification formalism is limited. Fixed point calculi seem to join the strong points of both logics and automata, since they provide an elegant and well–structured mathematical notation inducing nice semantical hierarchies; on the other hand, solutions to the relative algorithmic problems are usually already implicitly present in the structure of these calculi, as computing the least (or, dually, the greatest) fixed point is one of general paradigms of algorithms. It is the fixed point approach on which we shall focus our interest in the present paper.

In all fixed point calculi, the main feature of the syntax consists in an explicit notation for solutions of the fixed point equations $x = \tau(x)$. If x is supposed to range over a complete lattice and the operation defined by τ is monotone then, by the Knaster–Tarski Theorem, there exists a least fixed point $\mu x.\tau(x)$ and a greatest fixed point $\nu x.\tau(x)$. Here τ may depend on some arguments other than x that may be subjected to further applications of fixed point operators which by this can be nested and alternated, e.g. $\mu x.\nu y.\tau(x, y)$, $\nu x.\tau(x, \mu y.\tau_1(x, y))$, etc.

The role of *alternation* of least and greatest fixed point operators as a source of a sharp expressive power for the fixed point calculus has been recognized in early 1980'ies. D. Park in his studies on the semantics of parallelism [39] observed that what he considered as a *fair merge* of two infinite sequences[1] can be adequately characterized only using *both* extremal fixed point operators. Flon and Suzuki [15] and later Emerson and Clarke [8] gave insighting fixed point characterizations of several correctness properties of parallel programs including freedom from deadlock, invariance, and inevitability under fair scheduling assumptions. Arnold and Nivat [1] proposed the greatest fixed points as semantics for nondeterministic recursive programs, and Niwiński [34] has extended their approach to alternated fixed points in order to capture the infinite behavior of context–free grammars. Kozen [23] introduced the *modal μ–calculus*, which is a propositional modal logic extended by the least fixed point operator. This logic has subsequently received much studies; now it is known to subsume most of the known propositional logics of programs and yet to be decidable in the exponential time [9]; it also possess a natural complete proof system [54].

Another notation of μ–terms (called fixed point terms in the present paper) has been introduced by this author in [35, 36]; it simply extends the usual notation of first order terms by the least and greatest fixed point operators, and a binary operator $\vee$ as the only logical operator. Fixed point terms can be interpreted in algebras with completely ordered universes, under the assumption that the basic operations are monotonic (justification of this semantics comes from the Knaster–Tarski theorem, and no continuity requirements are needed). Important examples of such algebras can be obtained by the powerset construction applied to arbitrary algebraic structures. It turns out that a rather simple notation of fixed point terms, when interpreted in the powerset algebra of trees, has precisely the same expressive power as Rabin automata. Consequently, by the results of Rabin [41], it captures the expressive power of monadic second order logic over a tree structure which is known to be one of the most expressive among the decidable mathematical theories. The proof of this result, first announced in [37], will be presented in details in the present paper. Actually, the characterization will be proved on a more general level : rather than automata on trees, we shall consider automata running over arbitrary algebraic structures;

[1]By Park, the fair merge of two infinite sequences of 0's and of 1's is the set of sequences where both bits occur infinitely often. It can be characterized by the term $\nu y.\mu x.0x \vee 1x \vee 01y$.

this concept will subsume, in particular, alternating automata on trees. We would like to point out that a computation of an automaton may be infinite even if the underlying algebraic structure is finite. The aforementioned generalization will be useful especially in solving the nonemptiness problem for fixed point terms (that is, the question whether a term has a nonempty denotation), as it will allow to confine it to a particularly simple powerset algebra actually equivalent to the Boolean algebra $\{0,1\}$. This will induce a polynomial time algorithm for the problem. As for the powerset algebra of trees, it will be shown to play the role of an initial structure in the class of all powerset algebras.

A tight correspondence will be established between the hierarchy of indices of Rabin automata and a hierarchy of alternation of the least and greatest fixed point operators. It will be exhibited that both hierarchies may be finite or infinite varying from one interpretation to another. In particular, in a powerset algebra of trees constructed from function symbols of a given signature, the hierarchy is infinite if the signature contains at least two symbols of which at least one has the arity at least 2.

The paper is organized as follows. In the preliminary Section 1, we present basic concepts concerning trees, powerset algebras and automata. We introduce our concept of automata running in arbitrary algebras which is a straightforward generalization of automata on infinite trees. The subsequent Section 2 is an introduction to the calculus of fixed point terms. In Section 3, we prove a fine equivalence between the fixed point terms and Rabin automata. This result is completed by a discussion of an alternative characterization of the same hierarchy, by means of two iteration operators. Section 4 presents the aforementioned initiality result of the powerset algebra of trees and discusses the nonemptiness (or satisfiability) problem for fixed point terms. Section 5 is devoted to the hierarchy problem. Finally, in Section 6 we discuss relations between our fixed point calculus and other formalisms considered in the literature.

Notations

The set of natural numbers 0,1, ..., or, equivalently, the first infinite ordinal, is denoted by ω. For a set X, $\wp(X)$ is the *powerset* of X, i.e. the set of all subsets of X; X^n is the Cartesian product of n copies of X; $|X|$ is the *cardinality* of X. The symbol $\leq$ is used with several meanings: depending on the context, it may stand for the standard ordering of natural numbers (or ordinals), the initial segment orderings of words, as well as an abstract partial ordering in an abstract poset. Once the meaning of $\leq$ is understood, the symbols $\geq$, $<$, have their usual meaning.

Throughout the paper, we largely use vector notation: generally, a tuple $\langle a_1, \ldots, a_n \rangle$ may be abbreviated $\vec{a}$. This notation is also applied in more complex terms and formulas, e.g., in writing $\vec{t}[\vec{s}/\vec{x}]$ instead of $\langle t_1[s_1/x_1, \ldots, s_m/x_m], \ldots, t_n[s_1/x_1, \ldots, s_m/x_m] \rangle$, or $\vec{a} \leq \vec{b}$ to mean "$a_1 \leq b_1$ and ... and $a_n \leq b_n$", and also $\vec{X} \subseteq \vec{Y}$ to mean "$X_1 \subseteq Y_1$ and ... and $X_n \subseteq Y_n$".

1 Preliminaries

1.1 Trees

For a set X, X^* denotes the free monoid generated by X, i.e., the set of all finite words that can be written with X as an alphabet, including the empty word ϵ. A word $w \in X^*$ can be uniquely decomposed in $w = x_1 \cdots x_k$, where $x_1, \ldots, x_k \in X$; the number k is called the *length* of w, and denoted by $|w|$ (in particular $|\epsilon| = 0$). The *concatenation* of two words $w = x_1 \cdots x_k$ and $v = y_1 \cdots y_m$ is the word $x_1 \cdots x_k y_1 \cdots y_m$ denoted by wv. The similar notation will be applied to sets of words: if $L, K \subseteq X^*$, $LK = \{uv \mid u \in L, v \in K\}$. Note that $L\{\epsilon\} = \{\epsilon\}L = L$, while $L\emptyset = \emptyset L = \emptyset$.

The *initial segment* relation $\leq$ is defined by $u \leq w$ iff there exists v such that $uv = w$. Note that ϵ is the least element of X^* with respect to $\leq$.

Any nonempty subset T of X^* closed under initial segments is called a *tree*. The elements of T are usually called *nodes*. The word ϵ, which belongs to every tree, is called the *root*. The $\leq$-maximal elements of T are called *leaves*. If $u \in T$, $x \in X$, and $ux \in T$ then ux is an *immediate successor* of u in T. A finite or infinite sequence $w_0, w_1, \ldots, w_m, \ldots$ such that $w_0 = \epsilon$ and, for each m, whenever w_{m+1} is defined, it is an immediate successor of w_m, is called a *path* in T. A finite path $w_0, w_1, \ldots, w_m$ is *complete* if w_m has no immediate successors. We recall the celebrated *König's Lemma.*

> If $T \subseteq X^*$ is an infinite tree and each $w \in T$ has only a finite number of immediate successors in T then T has an infinite path.

A tree with no infinite path is called *well–founded.* We note a fundamental property of trees that we shall refer to as *Tree Induction Principle* (c.f., e.g., [26]).

> Let T be a well–founded tree and let S be a subset of T such that, for all $w \in T$, if *all* immediate successors of w are in S, so is w. Then $S = T$.

Now let S be an arbitrary set. If T is a tree then a mapping $t : T \to S$ is called an *S-valued tree* or shortly an S-tree; in this context T is called the *domain* of t and denoted by $T = \operatorname{dom} t$. We say "root of t", "path in t", etc., referring to the corresponding objects in $\operatorname{dom} t$. For a node $v \in \operatorname{dom} t$, the *subtree* of t induced by v is the S–tree denoted by $t.v$ and defined by

- $\operatorname{dom} t.v = \{w \mid vw \in \operatorname{dom} t\}$,
- $t.v(w) = t(vw)$, for $w \in \operatorname{dom} t.v$.

We shall be often interested in the set of values that occur infinitely often along a path in a tree. Let t be an S–tree and let $P = (w_0, w_1, \ldots)$ be an infinite path in t. We let

$$\operatorname{Inf}(t, P) = \{s \in S : t(w_m) = s \text{ for infinitely many } m\}$$

Observe that if S is finite then $\operatorname{Inf}(t, P)$ is always nonempty and there is some m_0, such that $(\forall m \geq m_0)\ t(w_m) \in \operatorname{Inf}(t, P)$.

A basic construction that is frequently used for trees is substitution. Suppose $A \subseteq \operatorname{dom}(t)$ is an *antichain* with respect to $\leq$ (i.e., any two elements of A are incomparable), and let f be a function associating an S-tree $f(w)$ with each $w \in A$. Then the *substitution* tree $t[f]$ is the S-tree defined by

- $\operatorname{dom} t[f] = \bigcup_{w \in A} w \operatorname{dom} f(w) \cup \{w \in \operatorname{dom} t : \forall w' \leq w, w' \notin A\}$,
- $t[f](u) = \begin{cases} f(w)(v) & \text{if } u = wv, w \in A \\ t(u) & \text{otherwise.} \end{cases}$

The above definition of substitution can be naturally extended to the case when f is a mapping associating a *set* of S-trees $f(w)$ with each $w \in A$; in this case the result of substitution is a set of S-trees. More specifically, we call f' a *choice function for* f if, for each $w \in A$, $f'(w)$ is a tree in $f(w)$. (Observe that a choice function may fail to exist if some $f(w) = \emptyset$.) We set

$$t[f] \quad =_{df} \quad \{t[f'] : f' \text{ is a choice function for } f\}$$

In the case A is finite, say $A = \{w_1, \ldots, w_k\}$, we shall often express f explicitly, writing for example $t[w_1 \leftarrow t_1, \ldots, w_k \leftarrow t_k]$, in vector notation $t[\vec{w} \leftarrow \vec{t}]$. This notation will be extended to sets of S-trees, say $L_1, \ldots, L_k$ by

$$t[w_1 \leftarrow L_1, \ldots, w_k \leftarrow L_k] \quad = \quad \{t[w_1 \leftarrow t_1, \ldots, w_k \leftarrow t_k] : t_1 \in L_1, \ldots, t_k \in L_k\}$$

We say that two infinite paths : $P = (w_0, w_1, \ldots)$ in t and $P' = (w'_0, w'_1, \ldots)$ in t' are *cofinal*, if, roughly speaking, from some moments on, "they look the same "; more specifically, if there exist a sequence of words $v_0, v_1, \ldots$, and $m_1, m_2 < \omega$, such that $w_{m_1+k} = w_{m_1} v_k$, $w'_{m_2+k} = w'_{m_2} v_k$, and moreover $t(w_{m_1+k}) = t(w'_{m_2+k})$, for $k < \omega$. Clearly, if P and P' are cofinal then $\operatorname{Inf}(t, P) = \operatorname{Inf}(t', P')$. Observe that, if $t[f]$ is a substitution tree, then every path in any $f(v)$ is cofinal with some path in $t[f]$. We shall often make implicit use of this and other easily–noted properties in the sequel.

We also introduce a concept of limit. Suppose $t_0, t_1, \ldots$ is a sequence of S-trees such that $\operatorname{dom} t_0 \subseteq \operatorname{dom} t_1 \subseteq \cdots$, and, for each $w \in \bigcup_{n<\omega} \operatorname{dom} t_n$, there is some $m(w)$ such that $\forall m \geq m(w)$, $t_m(w) = t_{m(w)}(w)$. Then we define the S-tree t by

- $\operatorname{dom} t = \bigcup_{n<\omega} \operatorname{dom} t_n$,
- $t(w) = t_{m(w)}(w)$, for $w \in \operatorname{dom} t$.

By hypothesis, t is well–defined. We call it the *limit* of the sequence t_n and denote it by $\lim t_n$.

Now let *Sig* be a finite *signature*, i.e., a finite set of function symbols, each $f \in Sig$ given with a finite arity $\rho(f) \geq 0$. The symbols of arity 0 are also called *constant symbols*. A *syntactic tree* over *Sig* is a *Sig*-tree $t : \operatorname{dom}(t) \to Sig$, where $\operatorname{dom}(t)$ is included in ω^* (recall that ω denotes the set of natural numbers) and the following condition is satisfied: if $t(w)$ is a symbol of arity k then w has exactly k immediate successors in $\operatorname{dom}(t)$ which are $w1, \ldots, wk$. Note that a node of a syntactic tree is a leaf if and only if it is labeled by a constant symbol. Clearly, the set of syntactic trees is closed under the substitution and limit operations.

The collection of all syntactic trees over *Sig* is denoted by T_{Sig}.

For syntactic trees, we shall often consider some special kind of substitution. Let $c_1, \ldots, c_m$ be some constant symbols in *Sig* and let $L_1, \ldots, L_m \subseteq T_{Sig}$. Let $t \in T_{Sig}$ and

$A = \{w \in \mathrm{dom}(t) : t(w) \in \{c_1, \ldots, c_m\}\}$. Clearly, A is an antichain in $\mathrm{dom}\, t$. Then we define $t[c_1 \leftarrow L_1, \ldots, c_m \leftarrow L_m]$ as $t[f]$, where f is mapping that sends each $w \in A$ with $t(w) = c_i$ onto L_i. This notation will be also extended to a set of trees $K \subseteq T_{Sig}$ by

$$K[c_1 \leftarrow L_1, \ldots, c_m \leftarrow L_m] \quad =_{df} \quad \{t[c_1 \leftarrow L_1, \ldots, c_m \leftarrow L_m] : t \in K\}$$

Remark. It is apparent and well–known that finite syntactic trees can be identified with closed terms over the signature *Sig*. Therefore, syntactic trees may be viewed as an infinitary extension of terms. This point of view has been developed in the theory of algebraic semantics of programs, see e.g.[16].

1.2 Algebras

We now present the kinds of algebraic structures to be considered in our study. An obvious requirement of models of fixed point calculi is the existence of definable fixed points. This is provided by a concept of a μ–algebra in which the universe is a complete lattice and the basic operations are monotonic. As we shall be interested in a fixed point characterization of automata operating on algebraic structures, our main focus will be on a class of μ–algebras that can be obtained by a powerset construction from arbitrary algebras or, more generally, from what we shall call semi–algebras.

Complete lattices

A partially ordered set $\langle L, \leq \rangle$ is a *lattice* if for any two elements $a, b \in L$, there exists their least upper bound in L, denoted $a \vee b$, and their greatest lower bound, $a \wedge b$. A lattice is said to be *complete*, whenever each subset A of L has a least upper bound in L; we denote it by $\bigvee A$. Note that consequently each $A \subseteq L$ has also a greatest lower bound, which is the least upper bound of the set of lower bounds of A, it will be denoted $\bigwedge A$. In particular L itself has extremal elements $\bigvee L$ and $\bigwedge L$, also denoted by $\top$ and $\bot$. A k–ary function $f : L^k \to L$ is said to be *monotonic* if $f(a_1, \ldots, a_k) \leq f(b_1, \ldots, b_k)$, whenever $a_i \leq b_i$ holds for all $i = 1, \ldots, k$.

Semi–algebras

Recall [18] that an algebra over a signature *Sig* (briefly: *Sig*-algebra), usually presented by $\mathcal{A} = \langle A, \{f^{\mathcal{A}} : f \in Sig\}\rangle$, consists of a set A called the *universe* of $\mathcal{A}$ and a family of (basic) *operations* of $\mathcal{A}$ which, for each $f \in Sig$, comprises an interpretation of f, that is a mapping $f^{\mathcal{A}} : A^{\rho(f)} \to A$.

(The convention that a Roman capital letter stands for a universe of a structure denoted by the corresponding calligraphic letter will be maintained in the sequel.)

A more general concept is that of a *partial algebra* [18], in which the operations need not to be everywhere defined. In our considerations, it is natural to introduce a yet more general notion of a semi–algebra over *Sig*, in which the operations may be also over–defined. A *semi–algebra* is a structure of the form $\mathcal{B} = \langle B, \{f^{\mathcal{B}} : f \in Sig\}\rangle$ where, for each $f \in Sig$ of arity $\rho(f)$, $f^{\mathcal{B}}$ is a $\rho(f)+1$ ary relation over B, that is, $f^{\mathcal{B}} \subseteq B^{\rho(f)+1}$. We shall write $b \doteq f(b_1, \ldots, b_{\rho(f)})$ to mean $(b_1, \ldots, b_{\rho(f)}, b) \in f^{\mathcal{B}}$. A (partial) algebra $\mathcal{A}$ can be viewed as a semi–algebra if we identify a function $f^{\mathcal{A}}$ with its graph.

Our two basic examples concern trees. It is well–known that the set of closed terms over Sig can be organized into an algebra which is a free algebra in the class of all algebras over Sig with the empty set of generators [18]. This structure can be further extended to an algebra of all syntactic trees over Sig that will be presented by $\mathcal{T}_{Sig} = \langle T_{Sig}, \{f^{\mathcal{T}_{Sig}} \mid f \in Sig\}\rangle$, where, for $f \in Sig$ and $t_1, \ldots, t_{\rho(f)} \in T_{Sig}$, $f^{\mathcal{T}_{Sig}}(t_1, \ldots, t_{\rho(f)})$ is the unique Sig-tree t satisfying the following conditions:

- $\mathrm{dom}\, t = \{\epsilon\} \cup 1\mathrm{dom}\, t_1 \cup \cdots \cup \rho(f)\mathrm{dom}\, t_{\rho(f)}$,
- $t(\epsilon) = f$,
- $t(iw) = t_i(w)$, for $i \in \{1, \ldots, \rho(f)\}$ and $w \in \mathrm{dom}\, t_i$.

On the other hand, any tree $t \in T_{Sig}$ can be itself considered as a semi–algebra $\mathbf{t} = \langle \mathrm{dom}(t), \{f^{\mathbf{t}} \mid f \in Sig\}\rangle$ such that for all $f \in Sig$, $f^{\mathbf{t}} = \{(w1, \ldots, w\rho(f), w) : t(w) = f\}$.

Note that in general $\mathbf{t}$ is not even a partial algebra (in the sense of [18]), since a constant symbol c may be interpreted as a set of nodes (those for which $t(w) = c$), rather than a single node.

We call a semi–algebra $\mathcal{B}$ *operationally complete* if any $b \in B$ is in the image of some operation, that is, there is some $f \in Sig$, and $b_1, \ldots b_{\rho(f)} \in B$, such that $b \doteq f^{\mathcal{B}}(b_1, \ldots, b_k)$ (if $\rho(f) = 0$, this condition means simply $b = f^{\mathcal{B}}$).

For technical reasons, it is convenient to make the following assumption.

Proviso 1

All semi–algebras considered in this paper are operationally complete.

Remark. It is natural to consider a mapping $h : A \to B$ a *homomorphism* from a semi–algebra $\mathcal{A}$ to a semi–algebra $\mathcal{B}$ if, for each $f \in Sig$ and $a_1, \ldots, a_{\rho(f)}, a \in A$, $(a_1, \ldots, a_{\rho(f)}, a) \in f^{\mathcal{A}}$ implies $(h(a_1), \ldots, h(a_{\rho(f)}), h(a)) \in f^{\mathcal{B}}$. This notion coincides with the usual concept of homomorphism if $\mathcal{A}$ and $\mathcal{B}$ are ordinary algebras. Then, for any tree $t \in T_{Sig}$, a mapping $h : \mathrm{dom}(t) \to T_{Sig}$, given by $w \mapsto t.w$ is a homomorphism from the semi–algebra $\mathbf{t}$ to the algebra $\mathcal{T}_{Sig}$.

μ–Algebras

A *μ–algebra* over signature Sig is a pair $\langle \mathcal{A}, \leq_{\mathcal{A}}\rangle$, where $\mathcal{A} = \langle A, \{f^{\mathcal{A}} : f \in Sig\}\rangle$ is a Sig–algebra and $\leq_{\mathcal{A}}$ is an ordering on A such that $\langle A, \leq_{\mathcal{A}}\rangle$ is a complete lattice and all the operations $f^{\mathcal{A}}$, $f \in Sig$, are monotonic.

It may be sometimes convenient to include the lattice operations $\vee$ and $\wedge$, or one of them, to the family of basic operations of a μ–algebra. We reserve the symbols $\vee$ and $\wedge$ for this purpose, they are always considered as binary and interpreted in the standard way, whenever $\mathcal{A}$ is a μ–algebra. If $\vee, \wedge \notin Sig$, we abbreviate the signature $Sig \cup \{\vee\}$ by $Sig_\vee$, $Sig \cup \{\wedge\}$ by $Sig_\wedge$, and $Sig \cup \{\vee, \wedge\}$ by $Sig_{\vee,\wedge}$.

If $\mathcal{A}$ is a μ–algebra over Sig, we define a μ–algebra $\mathcal{A}_\vee$ over $Sig_\vee$ as the enrichment of $\mathcal{A}$ by the operation $\vee^{\mathcal{A}}$, interpreted as the least upper bound in $\langle A, \leq_{\mathcal{A}}\rangle$. The μ–algebras $\mathcal{A}_\wedge$ and $\mathcal{A}_{\vee,\wedge}$ are defined similarly.

Powerset algebras

Let $\mathcal{B} = \langle B, \{f^{\mathcal{B}} : f \in Sig\}\rangle$ be a semi–algebra over signature Sig. The *powerset algebra* of $\mathcal{B}$ is an algebra over $Sig_\vee$ of the form

$$\wp\mathcal{B} = \langle \wp(B), \{\cup\} \cup \{f^{\wp\mathcal{B}} : f \in Sig\}\rangle$$

where, $\cup = \vee^{\wp\mathcal{B}}$ is the (binary) set union and, for each $f \in Sig$, and $L_1, \ldots, L_{\rho(f)} \subseteq B$,

$$f^{\wp\mathcal{B}}(L_1, \ldots, L_{\rho(f)}) = \{b : (\exists a_1 \in L_1, \ldots, \exists a_{\rho(f)} \in L_{\rho(f)})\, f^{\mathcal{B}}(a_1, \ldots, a_{\rho(f)}) \doteq b\}$$

Now the universe of $\wp\mathcal{B}$, $\wp(B)$, can be considered as a complete lattice with the subset ordering and clearly all the basic operations of $\wp\mathcal{B}$ are monotonic. Therefore, the pair $\langle \wp\mathcal{B}, \subseteq\rangle$ forms a μ–algebra; in the sequel, we shall identify the powerset algebra $\wp\mathcal{B}$ with this μ–algebra.

Our most important example of a powerset algebra will be the powerset algebra of the aforementioned algebra of syntactic trees over a signature Sig, in symbols $\wp\mathcal{T}_{Sig}$. If a signature is fixed in the context or irrelevant, we shall usually call this structure a *powerset tree algebra.*

Powerset algebras with intersection

By definition, $\wp\mathcal{B}_\wedge$ is an enrichment of $\wp\mathcal{B}$ by the lattice operation $\wedge$ that in the lattice $\langle \wp(B), \subseteq\rangle$ is the set intersection. This enrichment may be essential since $\cap$ need not be in general definable from other operations. We shall note, however, that it is redundant in the class of all powerset algebras, namely, that $\wp\mathcal{B}_\wedge$ can be identified with some $\wp\mathcal{B}'$, where $\mathcal{B}'$ is a suitable enrichment of $\mathcal{B}$.

To this end, we need a concept of isomorphism between μ–algebras. Let $\mathcal{A}$ and $\mathcal{A}'$ be μ–algebras over the same signature Sig. We say that a *bijective* mapping $h : A \to A'$ is an *isomorphism* from $\mathcal{A}$ to $\mathcal{A}'$, if

1. it is an isomorphism of lattices, that is $a \leq_{\mathcal{A}} b$ if and only if $h(a) \leq_{\mathcal{A}'} h(b)$, for any $a, b \in A$;
2. it preserves the basic operations, that is, $h(f^{\mathcal{A}}(a_1, \ldots, a_{\rho(f)})) = f^{\mathcal{A}'}(h(a_1), \ldots, h(a_{\rho(f)}))$, for all $f \in Sig$ and $a_1, \ldots, a_{\rho(f)} \in A$.

Clearly, in this case the inverse mapping $h^{-1} : A' \to A$ is an isomorphism from $\mathcal{A}'$ to $\mathcal{A}$, which justifies the saying that the μ–algebras $\mathcal{A}$ and $\mathcal{A}'$ are *isomorphic* .

Now let $\mathcal{B}$ be an arbitrary semi–algebra over a signature Sig. Without loss of generality we may assume that the symbol $\wedge$ is not in Sig. We shall consider a semi–algebra $\mathcal{B}'$ over the signature $Sig_\wedge$ such that $B' = B$ and $f^{\mathcal{B}'} = f^{\mathcal{B}}$, for all $f \in Sig$, while $\wedge^{\mathcal{B}'}$ is a (partial) operation defined by

$$\begin{array}{lcl} a \wedge a & = & a \\ a \wedge b & \text{undefined}, & \text{for } a \neq b \end{array}$$

Note that we have momentarily suspended our convention about the use of $\wedge$ since B is not a lattice and so $\wedge^{\mathcal{B}'}$ is not a greatest lower bound in any sense. However, if we proceed to the powerset algebra of $\mathcal{B}'$, we have obviously

$$L_1 \wedge^{\wp\mathcal{B}'} L_2 \quad = \quad L_1 \cap L_2$$

for any $L_1, L_2 \subseteq B' = B$. Hence, the following is immediate.

Proposition 1.1 *The μ–algebras (over signature* $\mathrm{Sig}_{\vee,\wedge}$*) $\wp\mathcal{B}_\wedge$ and $\wp\mathcal{B}'$ are isomorphic.*

1.3 Automata

We are going to present the concept of a finite state automaton which can have infinite yet successful computations. Traditionally automata of that kind are considered as running over infinite objects, e.g. infinite words or trees with infinite paths (c.f. *Historical note on automata* below). In the present framework, we allow an automaton to run over elements of an arbitrary semi–algebra. Classical automata on trees fit that concept as a typical case. We remark that a computation of an automaton may be infinite also if the underlying algebra is finite.

Informal description

Essentially an automaton consists of a set of transitions and a global acceptance condition. Transitions are equations of the form $x = f(x_1, \ldots, x_k)$, where $f \in Sig$. Here, $x, x_1, \ldots, x_k$ are the automaton's *states*. A computation of an automaton in a semi–algebra $\mathcal{B}$ can be described as a process running in discrete, possibly infinite time. Initially, one copy of the automaton examines some element of the semi–algebra assuming an initial state. At each moment, a number of elements of $\mathcal{B}$ will be examined by a number of copies (not excluding that many copies can examine the same element), but all these copies operate independently, without any communication. Suppose that a copy of the automaton examines an element $b \in B$ assuming a state x. Then it tries to *decompose* b according to some transition $x = f(x_1, \ldots, x_k)$, that is, to find some $b_1, \ldots, b_k \in B$, such that $b \doteq f^{\mathcal{B}}(b_1, \ldots, b_k)$. If this attempt succeeds, our copy splits into k new copies that are sent to the elements $b_1, \ldots, b_k$, assuming the states $x_1, \ldots, x_k$ respectively. Note that a transition is chosen nondeterministically, but once it is determined, *all* the new–born copies must appear. We note that a transition may have a form $x = f$ where f is a constant symbol, in this case decomposition succeeds if $b \doteq f^{\mathcal{B}}$ and, if so, this computation path successfully terminates.

Now if we follow an infinite path of the computation process, we can see an infinite sequence of states, some of them, due to finiteness of the automaton, reappearing infinitely often. Thus, with each infinite computation path, we can associate its *permanent* set of states. This is of which the *global acceptance condition* of the automaton will take care, by specifying which sets of states are *accepting* permanent sets. The acceptance paradigm is that the whole process is accepting iff *all* its paths are accepting, and, a given element b is accepted (or recognized) by the automaton if there *exists* an accepting process, in which b is examined initially. In this way an automaton defines a subset of B consisting of the accepted elements.

In view of subsequent applications, it will be useful to introduce a slight technical extension of the above–described concept, such that an automaton may define a k–ary operator $(\wp(B))^k \to \wp(B)$ rather than just an element of $\wp(B)$. To this end we equip an automaton with an additional feature called *variables*. A variable may occur on the right–hand side of a transition $x = f(x_1, \ldots, x_k)$ but not on the left–hand side. A computation process is defined w.r.t. a valuation of the variables by subsets of B. Now if a copy of automaton examines an element $b \in B$ assuming a variable z instead of a state, the computation path terminates in this point and it is successful iff b belongs to the set associated with z by the valuation. (The acceptance criterion for infinite paths remains unchanged.) Thus an automaton defines an operator from the set of possible valuations, which can be identified with some $(\wp(B))^k$, to $\wp(B)$.

We shall now give the formal definitions of the concepts described above. It is convenient to separate the "local" and "global" parts of automaton.

Pre–automata

A *pre–automaton* over a signature Sig can be presented as a tuple

$$A = \langle Sig, Q, V, Tr \rangle,$$

where Q is a finite set of *states*, V is a finite set of *variables*, and Tr is a set of *transitions*, $Tr \subseteq \bigcup_{f \in Sig} Q \times \{f\} \times (Q \cup V)^{\rho(f)}$.

Each transition is therefore a tuple $\langle y, f, x_1, \ldots, x_{\rho(f)} \rangle$ where $f \in Sig$, $y \in Q$, and $x_1, \ldots, x_{\rho(f)} \in Q \cup V$. We shall usually represent such a transition as an equation $y = f(x_1, \ldots, x_{\rho(f)})$. The variable y will be referred to as the *head* of the transition $y = f(x_1, \ldots, x_{\rho(f)})$. Note that if $\rho(f) = 0$, a transition has form $y = f$.

The semantics of the automaton is defined as follows. Let $\mathcal{B}$ be a semi–algebra. Let val be a valuation that associates a subset of B with any variable in V. A *run* of the pre–automaton A w.r.t. the valuation val is formally represented as a tree $r : \operatorname{dom} r \to B \times (Tr \cup V)$, with $\operatorname{dom} r \subseteq \omega^*$, satisfying the following conditions.

Let $w \in \operatorname{dom} r$ and suppose $r(w) = (b, \beta)$. Then,

- if β is a transition in Tr, say $y = f(x_1, \ldots, x_k)$, then w has k immediate successors in $\operatorname{dom} r$, $w1, \ldots, wk$, and $r(wi) = (a_i, \alpha_i)$, where
  - α_i is a transition with head x_i, if x_i is a state,
  - α_i is x_i, if x_i is a variable,
  - $b \doteq f^{\mathcal{B}}(a_1, \ldots, a_k)$
- if β is a variable in V, say z, then w is a leaf of r, and $b \in val(z)$.

With a run r, we associate two other trees over the same domain, that we will call *the element part* and *the state part* of r and denote by $r\uparrow_1$ and $r\uparrow_2$ respectively. The tree $r\uparrow_1$: $\mathrm{dom}\, r \to B$ is defined as the composition[2] $\pi_1 \circ r$, and the tree $r\uparrow_2$: $\mathrm{dom}\, r \to Q \cup V$ is defined by

$$r\uparrow_2(w) = \begin{cases} y & \text{if } \pi_2 \circ r(w) \text{ is a transition with head } y, \\ z & \text{if } \pi_2 \circ r(w) \text{ is a variable } z. \end{cases}$$

(Here π_i is the projection of the Cartesian product $B \times (Tr \cup V)$ on the ith component.)

A run r such that $r(\epsilon) = \langle b, q\rangle$ will be called a *q–run* of A on b.

Global acceptance conditions

An *automaton* over signature Sig is presented as a tuple

$$A = \langle Sig, Q, V, q_0, Tr, Acc\rangle,$$

where $\langle Sig, Q, V, Tr\rangle$, is a pre-automaton, $q_0 \in Q$ is an *initial* state, and Acc is an *accepting condition.*

As we have mentioned in our informal description of automata, an accepting condition is essentially a property of sets of states, so it can be always represented by a subfamily of $\wp(Q)$. For technical reasons, it is often convenient to consider more specific kinds of acceptance conditions that can be represented in more succinct way. We mention below four kinds of accepting conditions with which we associate the names of R.J.Büchi, D.Muller, M.O.Rabin, and A.W.Mostowski (see *Historical note on automata* below). According to the actual accepting condition, we shall refer to *Büchi, Muller, Rabin* and *Mostowski automata*, respectively. We will say simply 'automaton', if the acceptance condition is irrelevant to the considerations.

Now, for each of the aforementioned accepting conditions we give its representation and specify which subsets $X \subseteq Q$ are considered accepting.

- *Muller acceptance condition* is given by a family of sets of states, $Acc = \mathcal{F} \subseteq \wp(Q)$, and a set $X \subseteq Q$ is accepting if $X \in \mathcal{F}$.
- *Büchi acceptance condition* is given by a set of states, $Acc = F \subseteq Q$; a set $X \subseteq Q$ is accepting if $X \cap F \neq \emptyset$.
- *Rabin acceptance condition* is given by a collection of pairs of sets of states,

  $Acc = \{(L_1, U_1), \ldots, (L_n, U_n)\}$; and a set $X \subseteq Q$ is accepting if, for some $i \in \{1 \leq i \leq n\}$, $X \cap U_i \neq \emptyset$ while $X \cap L_i = \emptyset$.
- *Mostowski acceptance condition* is given by a mapping $rank : Q \to \omega$; and a set $X \subseteq Q$ is accepting if $\max\{rank(x) : x \in X\}$ is even.

Since an automaton is a pre–automaton with some additional features, the concept of a run extends to automata immediately. Let $\mathcal{B}$ be a semi–algebra, and let r be a run of A over $\mathcal{B}$ w.r.t. some valuation *val*. Let $P = (w_0, w_1, \ldots)$ be an infinite path in $\mathrm{dom}\, r$.

[2] Throughout the paper, composition of mappings $f : A \to B$ and $g : B \to C$ is denoted $g \circ f$.

Recall (Section 1.1) that $\mathrm{Inf}(r\uparrow_2, P)$ is the set of states that occur infinitely often along the path P in the state part of r. We consider the path P *accepting* if the set $\mathrm{Inf}(r\uparrow_2, P)$ is accepting according to the condition Acc.

Note that for the Büchi condition it means that some state of F will reappear infinitely often; for the Rabin condition it means that, for some i, some state of U_i will reappear infinitely often and all states of L_i will disappear after finite time; for the Mostowski condition it means that the highest *rank* of a state occurring infinitely often is even.

A *run* is considered *accepting* if *all* its infinite paths are accepting. An element $b \in B$ is *accepted* (or recognized) by the automaton A w.r.t. a valuation *val*, if there exists an accepting run r w.r.t. this valuation, such that $r\uparrow_1(\epsilon) = b$ and $r\uparrow_2(\epsilon) = q_0$. The set of all such elements will be denoted $A^{\mathcal{B}}[val]$.

We usually present the valuations explicitly, e.g. $z_1 \mapsto K_1, \ldots, z_m \mapsto K_m$ ($\vec{z} \mapsto \vec{K}$, for short), where $V = \{z_1, \ldots, z_m\}$ and $K_1, \ldots, K_m \subseteq B$. Thus, an automaton with m variables (given in some order) induces an m–ary operator on $\wp(B)$ that sends a tuple $\vec{K}$ onto $A^{\mathcal{B}}[\vec{z} \mapsto \vec{K}]$. We shall denote this operator by $A^{\mathcal{B}}[\vec{z}]$ or simply by $A^{\mathcal{B}}$. We shall also write $A(z_1, \ldots, z_m)$ to specify the variables of automaton A.

Rabin automata vs other kinds of automata

We say that two acceptance conditions Acc and Acc' over the same set of states Q although possibly of different kinds are *equivalent* if any set $X \subseteq Q$ is accepting according to Acc iff it is accepting according to Acc'. As we have already remarked, any Büchi, Rabin, or Mostowski condition can be presented as a Muller condition. Also, a Büchi condition given by a set of states F is obviously equivalent to a Rabin condition $\{(\emptyset, F)\}$.

It is also easy to see that the Mostowski condition given by a mapping $rank : Q \to \omega$ is equivalent to a Rabin condition of the form ($\{q : rank(q)$ is odd and $\geq i+1\}$, $\{q : rank(q)$ is even and $\geq i\}$), where i ranges over even numbers less than or equal to $\max\ rank\,(Q)$.

The last suggests a certain normal form of Rabin acceptance condition : we say that a Rabin condition is in the *chain form*, whenever it can be presented by $Acc = \{(L_1, U_1), \ldots, (L_n, U_n)\}$, where $L_1 \supseteq L_2 \supseteq \ldots \supseteq L_n$. We shall show that a Rabin condition as above is in turn equivalent to a Mostowski condition. Let the mapping *rank* be defined by

- $rank(q) = 2i$, for the greatest i such that $q \in U_i - L_i$, whenever such an i exists; otherwise :
- $rank(q) = 2i+1$, for the greatest i such that $q \in L_i$, whenever such an i exists; otherwise :
- $rank(q) = 1$.

We shall verify that both conditions are equivalent. Suppose that $X \subseteq Q$ is accepting according to the Rabin condition. Let i be the greatest one such that $X \cap L_i = \emptyset$ and $X \cap U_i \neq \emptyset$, and let $q \in X \cap (U_i - L_i)$. Then $rank(q)$ is even and $rank(q) \geq 2i$, while no state q' in X may have an odd rank $rank(q') > 2i$. Conversely, if $\max\,\{rank(x) : x \in X\} = 2i$, it is easy to see that X is accepting according to the pair (L_i, U_i).

We shall refer to Rabin automata with a condition in chain form, briefly, as to *Rabin chain automata.* Thus, Mostowski automata and Rabin chain automata are equivalent up to easy translations.

Note : Thomas [50] considers an apparently more restrictive form of a chain condition; in our notation it would amount to the requirement that the sets $L_n, U_n, L_{n+1}, U_{n+1}, \ldots, L_1, U_1$ form an increasing chain. However, whenever $L_1 \supseteq L_2 \supseteq \ldots \supseteq L_n$, the condition $\{(L_1, U_1), \ldots, (L_n, U_n)\}$ can be transformed to such a form in two steps. First, it is equivalent to $\{(L_1 \cup U_2 \cup \ldots \cup U_n, U_1), \ldots (L_{n-1} \cup U_n, U_{n-1}), (L_n, U_n)\}$. Next, we can replace each pair (X, Y) by $(X, X \cup Y)$.

We say that two automata A and A' are (semantically) *equivalent* if, for any semi–algebra $\mathcal{B}$, they define the same operator, that is, $A^{\mathcal{B}} = A'^{\mathcal{B}}$. Clearly, if two acceptance conditions Acc and Acc' over Q are equivalent then, for any pre–automaton A with the set of states Q, the automata (A, Acc) and (A, Acc') are equivalent. However, there exist more subtle relationships between different kinds of automata, usually involving transformation of the whole automaton rather than merely the acceptance condition. We summarize the known facts below (they have been originally proved for automata on k–ary trees, but the proofs adapt without difficulty to our automata as well).

Theorem 1.2 (Rabin [41]) *Any Muller automaton is equivalent to some Rabin automaton.*

Theorem 1.3 (Mostowski [28]) *Any Muller automaton is equivalent to some Mostowski automaton.*

In contrast :

Theorem 1.4 (Rabin [42]) *Let* $\mathrm{Sig} = \{a, b\}$, *with* $\rho(a) = \rho(b) = 2$.

The set of trees $t \in T_{Sig}$*, such that, on each path in* t*,* b *occurs only finitely often is recognizable by a Rabin automaton, but not by any Büchi automaton.*

Transformation from Muller to Rabin automata is rather standard, although it may increase the size of an automaton exponentially. Transformation from Muller to Mostowski automata originally given in [28] is double exponential, however Thomas [50] gives an elegant construction that increases the number of states from n to at most $n!$.

It should be noted that the three kinds of equivalent automata apparently differ in succinctness, and consequently also in the complexity of the emptiness problem. The emptiness problem for Rabin automata over infinite trees has been proved NP–complete by Emerson and Jutla [9], while the only upper bound known to us for the similar problem for Muller automata is double exponential. The emptiness problem for Mostowski automata is in *NP* $\cap$ *co–NP*, and it is a fascinating hypothesis that it may actually be in P. By contrast, the emptiness problem for Büchi automata is logspace complete for *PTIME* [42, 51].

In this paper, we primarily consider automata with the Rabin acceptance condition. This choice may be questioned, as the Muller condition is more general mathematically, while the Mostowski condition is quite simple and would make some proofs more elegant. On the other hand, the Rabin condition is based on the fundamental temporal property *some event repeats infinitely often*. We believe that if someone wishes to use automata for modeling behavior of a continuously operating parallel system, e.g., an e–mail net, she will rather use the Rabin acceptance condition. In our characterization of the computations of an automaton by fixed point operators, we shall see that the two basic temporal properties used in the Rabin condition : 'repeating infinitely often' and 'repeating only finitely often' are precisely captured by the greatest and the least fixed point operators respectively; this correspondence would not be apparent if we have chosen another acceptance condition.

Index of Rabin automaton

For a Rabin automaton with an acceptance condition of the form $\{(L_1, U_1), \ldots, (L_n, U_n)\}$, with $n \geq 0$, the number n is called the *index* of the automaton.

For our investigations of the expressive power of Rabin automata, it will be useful to refine this concept of index, taking into account that some acceptance pairs provide, in a sense, weaker restrictions than others. Indeed, there are two reasons for this. A pair of the form $(\emptyset, U_i)$ imposes only a positive constraint on the set of repeating states : some states have to reappear infinitely often. On the other hand, if for some pair (L_i, U_i), $L_i \cup U_i$ equals to the set of *all* states of the automaton then the constraint is purely negative : indeed, any path that does not meet the states from L_i infinitely often, is accepting according to this pair[3]. Note that a pair $(\emptyset, Q)$ gives no constraint at all.

These observations motivate the following definition.

We say that a Rabin automaton has index n *weakened by* $\emptyset$, if it has index n and the acceptance condition contains at least one pair of the form $(\emptyset, U)$. We also say that an automaton has index n *weakened "by all"*, if the acceptance condition contains (at least) one pair (L, U) such that $L \cup U = Q$.

Note that an automaton can have index n weakened both by $\emptyset$ and "by all" if the acceptance condition contains either two pairs corresponding to the both restrictions or a pair $(\emptyset, Q)$, where Q is the set of all states (for an index $n = 1$, the latter is the only possibility). Also note that an automaton with empty acceptance condition has the index 0 which, by definition, may not be weakened by anything.

Proviso 2

In order to avoid some tedious exceptions, we shall henceforth assume that for any automaton A in consideration, for any state x of A, there is always at least one transition of A with the head x (that is, in a form $x = \tau$). Clearly any automaton can be transformed to an automaton satisfying this assumption by adding some dummy states, although in one case this may require the change of the automaton's index, namely when this index is 1 weakened both by $\emptyset$ and "by all".

[3]It is not required in the definition of Rabin acceptance condition that $L_i \cap U_i = \emptyset$, but it should be clear that a pair (L_i, U_i) is semantically equivalent to $(L_i, U_i \setminus L_i)$. In particular, a pair (L_i, U_i) with $L_i \cup U_i = Q$ is equivalent to (L_i, Q).

Remark. We note that Proviso 2 can be in a sense viewed as a special case of Proviso 1 (page 7). Indeed, a pre–automaton $A = \langle Sig, Q, V, Tr\rangle$ may be viewed as a semi–algebra over signature $Sig \cup V$ (with $\rho(z) = 0$, for $z \in V$), the universe of which consist of $Q \cup V$ and the operations are defined by $x \doteq f^A(x_1, \ldots, x_{\rho(f)})$ iff $x = f(x_1, \ldots, x_{\rho(f)})$ is a transition in Tr, and $z^A = z$, for $z \in V$. Then our Proviso 2 means that this algebra is operationally complete.

Automata on trees

An automaton $A = \langle Sig, Q, V, q_0, Tr, Acc\rangle$ can, in particular, run in the algebra of trees $\mathcal{T}_{Sig}$. Let $val : V \to \wp(T_{Sig})$ be a valuation and let $t \in T_{Sig}$. Consider a run r of A w.r.t. the valuation val which initially examines the tree t. Since any subtree of t induced by a node w can be uniquely decomposed in $\mathcal{T}_{Sig}$ by $t.w = f^{\mathcal{T}_{Sig}}(t.w1, \ldots, t.w\rho(f))$, where $t(w) = f$, it can be easily seen that $\mathrm{dom}\, r \subseteq \mathrm{dom}\, t$ and moreover the element part of the run satisfies $r \uparrow_1 (w) = t.w$, for each $w \in dom\, r$. (For an automaton without variables, the domains $\mathrm{dom}\, r$ and $\mathrm{dom}\, t$ coincide.) Thus the element part of the run is redundant in this case as it can be all retrieved from $r \uparrow_1 (\epsilon)$.[4] This leads us to a simplified definition of a run in algebra of trees, which is the classical definition with only a slight modification due to the presence of variables :

> A run of an automaton r on a tree t w.r.t. a valuation val is a tree $r : \mathrm{dom}\, r \to Q \cup V$, such that $\mathrm{dom}\, r \subseteq \mathrm{dom}\, t$ and, for each $w \in \mathrm{dom}\, r$,
>
> (1) if $r(w) \in Q$ and $t(w) = f$ with $\rho(f) = k$, then $w1, \ldots, wk$ are in $\mathrm{dom}\, r$ and $r(w) = f(r(w1), \ldots, r(wk))$ is a transition in Tr;
>
> (2) if $r(w) \in V$, say $r(w) = z$, then w is a leaf of r and $t.w \in val\ (z)$.

Clearly, if r is a q–run on a tree t in a former sense then the state part of r, $r \uparrow_2$, is a q–run on t in the classical sense. Conversely, any run in the classical sense can be extended to a run in our sense in obvious way. Moreover, these transformations preserve the satisfaction of global acceptance conditions. Therefore, for the algebra of syntactic trees, our semantics of automata coincides with the classical one.

Alternating automata. Muller and Schupp [32] extended the concept of nondeterministic automata on trees to alternating automata. There are several equivalent ways of introducing this notion [29, 13]. Close to the our setting, we can present an *alternating automaton* by a tuple $A = \langle Sig, Q, V, q_0, Tr, Acc\rangle$ as above, where we additionally allow the set Tr to contain transitions of the form

$$x \;\; = \;\; y \wedge z$$

where $x \in Q$ and $y, z \in Q \cup V$. The semantics can be defined in game–theoretic terms. Let $t \in T_{Sig}$ and a valuation $val : V \to \wp(T_{Sig})$ be given. Consider a game of two players, say *Automaton* and *Opponent*. The game will consist of a finite or infinite sequence of plays; a position of a play will consist of a pair (w, x), where w is a node in $\mathrm{dom} t$, and

[4] We could alternatively consider a run of A in the semi–algebra $\mathbf{t}$ as defined on the page 7, initially examining the root of t. In this case, we would have simply $r \uparrow_1 (w) = w$, for $w \in \mathrm{dom}\, r$.

x is an item in $Q \cup V$. The initial position is (ϵ, q_0). If an actual position is (w, y) with $y \in Q$ then *Automaton* makes move by choosing a transition in Tr with a head y; the transition may be of a form $y = f(x_1, \ldots, x_k)$, where $t(w) = f$, or $y = x \wedge z$. In the former case, if moreover f is a constant symbol (that is, $k = 0$) then the game is over and *Automaton* wins. Otherwise *Opponent* answers by choosing some $i \in \{1, \ldots, k\}$; then the next position becomes (wi, x_i). If the transition chosen by *Automaton* was $y = x \wedge z$ then *Opponent* selects one of the conjuncts and then the next position is (w, x) or (w, z) according to this choice (note that the node part of the position has not changed). [5] If the actual position was (w, z) with $z \in V$ then the game is over and *Automaton* wins iff $w \in val(z)$. If the game is played infinitely then an infinite sequence of positions is selected. In this case, *Automaton* wins iff the set of variables that occur infinitely often at these positions satisfies the acceptance condition of A, otherwise, *Opponent* is the winner.

The set accepted by A under a valuation *val* consists of those trees $t \in T_{Sig}$ for which *Automaton* has a winning strategy in the game described above.

On the other hand, an automaton A presented above can be viewed as our automaton over the signature $Sig_\wedge$, and we can consider a computation of this automaton in the semi–algebra $\mathcal{T}'_{Sig}$ obtained as an enrichment of $\mathcal{T}_{Sig}$ by the interpretation of $\wedge$ given by $t \wedge t = t$ and $t \wedge t'$ undefined for $t \neq t'$ (Section 1.2, page 9).

Now, it is straightforward to see that these two semantics coincide, that is, a tree t is accepted by automaton A interpreted in $\mathcal{T}'_{Sig}$ if and only if it is accepted by the same A considered as an alternating automaton with the game–theoretic semantics defined above. Indeed, an accepting run of A in $\mathcal{T}'_{Sig}$ induces a winning strategy for *Automaton* and, conversely, from a winning strategy, an accepting run can be easily reconstructed.

Thus the concept of alternating automata on trees is subsumed by our notion of automata over arbitrary semi–algebras.

We shall not consider in this paper *alternating* automata over arbitrary semi–algebras. It can be noted, however, that the game semantics described above readily applies to our automata over semi–algebras as well. Since an arbitrary semi–algebra can be enriched by the operation $\wedge$ as above, it can be argued that our concept of automata captures already some essential features of alternation (nondeterminism and running parallel computations).

Elimination of variables. At this point we would like to make an observation that the operators over $\wp(T_{Sig})$ computed by (ordinary) automata with variables can be also characterized by tree languages computed by automata without variables, but over extended signature. More specifically, given an automaton $A(z_1, \ldots, z_m) = \langle Sig, Q, \{z_1, \ldots, z_m\}, q_0, Tr, Acc \rangle$, let us consider an automaton A' over a signature $Sig' = Sig \cup \{z'_1, \ldots, z'_m\}$, where the z'_i's are some fresh symbols and $\rho(z'_i) = 0$, for $i = 1, \ldots, m$, defined by $A' = \langle Sig', Q \cup V, \emptyset, q_0, Tr', Acc \rangle$, where the set of transitions Tr' is obtained from Tr by adding the equations $z_i = z'_i$, for $i = 1, \ldots, m$. Note that the automaton A' has no variables.

The following fact can be easily derived from the definitions.

[5] In this setting, the players may stay at the same node for ever, which is usually not allowed in a classical alternating automaton. It can be easily prevented by suitable restrictions on transition table of the automaton.

Proposition 1.5 *For any* $K_1, \ldots, K_m \subseteq T_{Sig}$,

$$L(A)[z_1 \mapsto K_1, \ldots z_m \mapsto K_m] \quad = \quad L(A')[z'_1 \leftarrow K_1, \ldots, z'_m \leftarrow K_m]$$

□

Examples of automata

1.Let $Sig = \{f, c\}$, where f is binary and c is a constant symbol. The following Büchi automaton accepts the set of trees that posses an infinite path on which infinitely many nodes have a left successor labeled by c :

$$\begin{aligned} Q &= \{ \text{ search, don't–care, guess , check } \} \\ q_0 &= \text{ search} \\ F &= \{ \text{ guess } \} \end{aligned}$$

and the set of transitions Tr consists of the following equations (written in the Backus–Naur form) :

$$\begin{aligned} \text{search} &= f(\text{ search , don't–care }) \,|\, f(\text{ don't–care , search }) \,| \\ &\quad\; f(\text{ don't–care , guess }) \,|\, f(\text{ guess , don't–care }) \\ \text{guess} &= f(\text{ check , search }) \\ \text{check} &= c \\ \text{don't–care} &= f(\text{ don't–care , don't–care }) \,|\, c \end{aligned}$$

2. Let a, b, c be binary symbols and let L be the set of trees over $\{a, b, c\}$, such that, on each path, both a and b occur infinitely often and c only finitely often. Then L is accepted by a Muller automaton with the states seen$_a$, seen$_b$, seen$_c$ (any of which can be initial state) and the transitions

$$\text{seen}_x \quad = \quad d(\text{ seen}_d, \text{ seen}_d)$$

for $x, d \in \{a, b, c\}$, and with the acceptance condition $\mathcal{F} = \{\{\text{ seen}_a, \text{ seen}_b\}\}$.

Note that replacing this Muller acceptance condition by a Rabin condition $\{(\{\text{ seen}_c\}, \{\text{ seen}_a, \text{ seen}_b\})\}$ would not lead to an equivalent automaton, since the last does not force that *both* a and b occur infinitely often along a path. The set L can be however recognized by a Rabin automaton, obtained from the above by adding a state success, and the transitions

$$\begin{aligned} \text{seen}_a &= b(\text{ success, success}) \\ \text{seen}_b &= a(\text{ success, success}) \\ \text{success} &= d(\text{ seen}_d, \text{ seen}_d) \end{aligned}$$

for $d \in \{a, b, c\}$. The acceptance condition of this automaton will be $\{(\{\text{ seen}_c\}, \{\text{ success}\})\}$.

3. Consider an automaton with one state x and two variables Y, Z, given by the transitions

$$x \quad = \quad a(x, x) \,|\, b(Y, Z)$$

and the trivial Büchi condition $F = \{x\}$. Then this automaton defines a binary operator over $\wp\mathcal{T}_{Sig}$ that sends a couple (K, M) on the set of trees $t \in \mathcal{T}_{Sig}$ such that, for each path, if b occurs the first time on the path, say at a node w, then the subtrees $t.w1$ and $t.w2$ belong to K and M respectively.

4. Let $A = \langle Sig, Q, q_0, Tr, Acc \rangle$ be an automaton (without variables). Following a remark after Proviso 2 (page14), the set Q can be organized into a semi–algebra over Sig (let us call it also A), such that $x \doteq f^A(x_1, \ldots, x_{\rho(f)})$ iff $x = f(x_1, \ldots, x_{\rho(f)})$ is a transition in Tr. Any q–run r of A in any semi–algebra B, on any element b, induces a q–run of A in the semi–algebra A, on the element q. (The state part of this new run coincides with that of r, and the element part coincides with the state part.) On the other hand, it is easy to see that any q–run of A in B also induces a q–run of A in $\mathcal{T}_{Sig}$, on some tree in T_{Sig}. Then, the following conditions are equivalent : (i) there exists an accepting q–run of A in some semi–algebra $\mathcal{B}$, (ii) there exists an accepting q–run of A in the semi–algebra A on q, (iii) there exists an accepting q–run of A in $\mathcal{T}_{Sig}$.

Historical note on automata

A comprehensive survey of automata on infinite objects is given Thomas [49]; an essay by Emerson [7] discusses the role of Büchi's ideas in computer science. Here we only briefly recall some points relevant to our considerations.

Infinite computations of finite state automata have been considered for the first time by Büchi [5] and Muller [31]. Considering such automata as running over infinite strings and taking as an acceptance criterion that a successful state should reappear infinitely often, Büchi proved the basic closure properties of these automata which allowed him to characterize the formulas of monadic second order logic interpreted in ω with (only) successor operation (see Section 6 below) and to establish decidability of this theory. Roughly speaking, one can say that any formula of monadic second order logic over ω describes a set of infinite computations of some automaton (interestingly, a similar paradigm appears in finite model theory). Rabin [41] proved an analogous result for the monadic second order theory of a tree with k successor operations, but now the closure under complement was a very difficult result, involving deep mathematical ideas. Also, the Büchi's acceptance condition turned out to be too weak; Rabin used a more general condition proposed by Muller [31] and also its restricted but sufficient form that we now refer to as the Rabin condition. Several authors attempted to simplify the Rabin's proof. Gurevich and Harrington [17] gave a proof *via* a rather difficult determinacy result for some infinite games with finite memory, with automaton as one of the players. In search of a yet more comprehensive argument, A.W. Mostowski was looking for automata capable to play with no memory at all; he succeeded with a concept of automata with a *chain acceptance condition* that, up to our knowledge, had been for the first time introduced by Klaus Wagner [52] (see also [53]) in context of automata on infinite words. Mostowski proved already in 1985 [28] that tree automata with this condition have the same accepting power as all Muller automata, and in [29] gave his proof of the Rabin complementation lemma. Incidentally, essentially the same acceptance criterion together with its *memory–less* property, has been later rediscovered by other authors, namely by McNaughton [30], under the name of a "split–free" Muller condition[6], and by Emerson and Jutla [13], under the name of a *parity acceptance condition*. The latter authors have also pointed out the relation between this condition and the fixed point calculus. In the present paper we have adopted the formulation due to Emerson and Jutla, but we think it justifiable to refer to the condition by the name of Mostowski who was the first to adopt the condition for automata on trees.

A generalization of the concept of automata to arbitrary algebras (not yet semi–algebras) has been proposed by Niwiński in [36]; a related concept of alternating automata running over transition systems has been considered recently by Janin and Walukiewicz [21].

2 Fixed point calculus

We first recall basic concepts and results concerning the least and greatest fixed points of monotonic mappings over complete lattices. Two issues are highlighted : the *vectorial* fixed points, which are conceptually very natural and technically useful, and a hierarchy induced by *alternation* of the least and greatest fixed point operators, which provides a natural measure of complexity of fixed point definable objects. Then, we introduce a logical formalism intended to deal with fixed point definable operations.

2.1 Basic results

Let $\langle L, \leq \rangle$ be a *complete lattice* (c.f. Section 1.2). The following basic result is due to Knaster [22] and Tarski [48].

Theorem 2.1 (Knaster–Tarski) *Any monotonic mapping $f : L \to L$ has a least fixed point*

$$\mu x.f(x) = \bigcap \{a \in L \mid f(a) \leq a\}$$

and a greatest fixed point

$$\nu x.f(x) = \bigcup \{a \in L \mid a \leq f(a)\}.$$

Proof. We shall show the second part of the result. Let r be $\bigcup\{a \in L \mid a \leq f(a)\}$. If b is a fixed point of f then $b \leq r$. It remains to verify that $r = f(r)$. Inequalities $a \leq f(a)$ and $a \leq r$ imply, by monotonicity of f, $a \leq f(r)$. Thus $r \leq f(r)$. Further, $f(r) \leq f(f(r))$, and then $f(r) \leq r$. $\square$

A well–known representation of extremal fixed points is provided by the notion of transfinite iterations. We shall not need this representation in further consideration, but we note it (without proof) for the sake of completeness.

Let $f : L \to L$ be a monotonic function. Define inductively the transfinite sequences $f^\xi(\bot)$ and $f^\xi(\top)$, where ξ is an ordinal number, by $f^0(\bot) = \bot$, $f^{\xi+1}(\bot) = f(f^\xi(\bot))$, and $f^\xi(\bot) = \bigcup\{f^\eta(\bot) \mid \eta < \xi\}$, if ξ is a limit ordinal; $f^0(\top) = \top$, $f^{\xi+1}(\top) = f(f^\xi(\top))$, and $f^\xi(\top) = \bigcap\{f^\eta(\top) \mid \eta < \xi\}$, if ξ is a limit ordinal.

[6]This is a Muller condition $\mathcal{F}$, such that if two sets X, Y both are (are not) in $\mathcal{F}$, so is (is not) $X \cup Y$.

Theorem 2.2 *There exist ordinals* α *and* α' *such that*

$$\mu x.f(x) = f^{\alpha}(\bot)$$

and

$$\nu x.f(x) = f^{\alpha'}(\top).$$

Remark. The reader may have noticed the symmetry between the properties of the least and the greatest fixed points. An obvious justification of this fact comes from consideration of the dual lattice $\langle L, \leq^* \rangle$ of $\langle L, \leq \rangle$, where $\leq^* = \geq$. Clearly, a mapping $f : L \to L$ is monotonic in the latter lattice iff it is monotonic in the former one, hence the least fixed point of f in $\langle L, \leq \rangle$ is the greatest fixed point in $\langle L, \leq^* \rangle$, and vice versa.

Combinatorial properties of fixed points

If $\langle L_1, \leq_1 \rangle, \ldots, \langle L_m, \leq_m \rangle$ are complete lattices so is the Cartesian product $L_1 \times \ldots \times L_m$ with the product ordering $(a_1, \ldots, a_m) \leq (b_1, \ldots, b_n)$ iff $a_i \leq_i b_i$, for $i = 1, \ldots, m$. We call a mapping with the domain $L_1 \times \ldots \times L_m$ monotonic if it is monotonic w.r.t. the product ordering. In particular a mapping $g : L^n \to L$ is monotonic if, for any $a_1, \ldots, a_n, b_1, \ldots, b_n$, $\vec{a} \leq \vec{b}$ implies $g(\vec{a}) \leq g(\vec{b})$. The monotonic mappings $f : L^n \to L$ will be called *operations* over L. Suppose h is a monotonic mapping from $L \times K$ to L and fix an element of K, say a, as a second argument of h. Then we obtain a monotonic mapping $h(x, a) : L \to L$. We can further consider the mapping from K to L which sends a on the least fixed point of $h(x, a)$; we shall denote it by $\mu x.h(x, y)$; the mapping $\nu x.h(x, y)$ is defined similarly. It is easy to see that these two mappings are again monotonic. In this context μ and ν can be viewed as operators that, given an operation on L, $f : L^n \to L$, produce new operations such as $\mu x_1.f(x_1, \ldots x_n)$, $\nu x_3.\mu x_2.\nu x_1.f(x_1, \ldots, x_n)$.

Remark. A reader that is not yet at ease with the above notation for nested fixed points, may find it helpful to think in terms of solving equations. Let, for example $f : L^3 \to L$ be a monotonic mapping over a complete lattice L. For fixed $\mathrm{x}_2, \mathrm{x}_3 \in L$, an equation $x_1 = f(x_1, \mathrm{x}_2, \mathrm{x}_3)$ has the least and the greatest solution in L. Then $\nu x_1.f(x_1, x_2, x_3)$ denotes the mapping that sends a couple $(\mathrm{x}_2, \mathrm{x}_3)$ onto the greatest solution of the above equation. Now, for fixed x_3, consider an equation $x_2 = \nu x_1.f(x_1, x_2, \mathrm{x}_3)$. Then $\mu x_2.\nu x_1.f(x_1, x_2, x_3)$ denotes the least solution of this last equation as a function of argument x_3. Finally, $\nu x_3.\mu x_2.\nu x_1.f(x_1, x_2, x_3)$ denotes the greatest solution of the equation $x_3 = \mu x_2.\nu x_1.f(x_1, x_2, x_3)$.

We recall some basic properties already noted in [35].

Proposition 2.3 *If* $g(x, y)$ *is an operation then*

$$\mu x.\mu y.g(x, y) = \mu x.g(x, x)$$

similarly for ν.

Proof. Let $a = \mu x.\mu y.g(x,y)$, $b = \mu x.g(x,x)$. We have $a = \mu y.g(a,y) = g(a, \mu y.g(a,y)) = g(a,a)$, hence $b \leq a$. To prove $a \leq b$, by 2.1 it is enough to show $\mu y.g(b,y) \leq b$. This follows from the fact that $b = g(b,b)$. □

Proposition 2.4 *If $g(x,y)$ is an operation then*

$$\mu x.\nu y.g(x,y) \leq \nu y.\mu x.g(x,y).$$

Proof Let $a = \mu x.\nu y.g(x,y)$. We have $a = g(a,a)$ and hence $\mu x.g(x,a) \leq a$. By monotonicity of the operation $\nu y.g(z,y)$ (with respect to z), we infer

$$\nu y.g(\mu x.g(x,a),y) \leq \nu y.g(a,y) = a$$

(the last equality follows from the definition of a).

Then, by monotonicity of g, we obtain

$$g(\mu x.g(x,a), \nu y.g(\mu x.g(x,a),y)) \leq g(\mu x.g(x,a),a)$$

and, by reducing the both sides,

$$\nu y.g(\mu x.g(x,a),y) \leq \mu x.g(x,a)$$

By Knaster-Tarski Theorem 2.1, this last inequality implies $\mu x.\nu y.g(x,y) \leq \mu x.g(x,a)$, with the left hand side equal a. Then, again by 2.1, $a \leq \nu y.\mu x.g(x,y)$, as required. □

Fixed point clones

Let $\langle L, \leq \rangle$ be a complete lattice. For $n \geq 0$ and $i \leq n$, let $\pi_n^i : L^n \to L$ be the ith *projection* of L^n on L, i.e. $\pi_n^i : \langle a_1, \ldots, a_n \rangle \mapsto a_i$.

A family of operations on L, say $\mathcal{C}$, is called a *clone* if it contains all the projections and is closed under composition, that is, if $f(x_1, \ldots, x_n), g_1(\vec{y}), \ldots, g_n(\vec{y})$ are in $\mathcal{C}$, so is $f(g_1(\vec{y}), \ldots, g_n(\vec{y}))$.

A family $\mathcal{C}$ is a *μ-clone* if it is a clone and moreover is closed under the μ operator, that is, if $f(x_1, \ldots, x_n)$ is in $\mathcal{C}$, so is $\mu x_i.f(x_1, \ldots, x_n)$, for $i = 1, \ldots, n$. A *ν-clone* is defined analogously. Finally, a family $\mathcal{C}$ is a *fixed point clone* if it is both a μ- and ν–clone.

For a family of operations $\mathcal{F}$ and $\eta \in \{\mu, \nu\}$, the (η-) clone *generated by* $\mathcal{F}$ is the intersection of all (η-) clones containing $\mathcal{F}$, which is obviously also a (η-) clone.

We denote the clone generated by a family $\mathcal{F}$ by Comp($\mathcal{F}$), the μ-clone generated by $\mathcal{F}$ by $\mu(\mathcal{F})$, the ν-clone generated by $\mathcal{F}$ by $\nu(\mathcal{F})$, and the fixed point clone generated by $\mathcal{F}$ by $fp(\mathcal{F})$. Note that $fp(\mathcal{F})$ is the closure of $\mathcal{F}$ under composition and under the least and the greatest fixed point operators.

It follows from the definitions :

Proposition 2.5

$$\mu(\mathcal{F}) = \mu(\mu(\mathcal{F})) = \mu(Comp(\mathcal{F})) = Comp(\mu(\mathcal{F})),$$
$$\nu(\mathcal{F}) = \nu(\nu(\mathcal{F})) = \nu(Comp(\mathcal{F})) = Comp(\nu(\mathcal{F})).$$

The fixed point clone generated by a given family of operations can be organized into a *hierarchy* according to the number of alternations of μ and ν. Adopting notation from the recursion theory, we form the classes Σ_n^μ and Π_n^μ, $n \geq 0$ as follows (here the superscript 'μ' stands as a decoration) :

$$\Sigma_0^\mu(\mathcal{F}) \;=\; \Pi_0^\mu(\mathcal{F}) \;=\; \mathrm{Comp}(\mathcal{F})$$

$$\begin{aligned} \Sigma_{n+1}^\mu(\mathcal{F}) &= \mu(\Pi_n^\mu(\mathcal{F})) \\ \Pi_{n+1}^\mu(\mathcal{F}) &= \nu(\Sigma_n^\mu(\mathcal{F})) \end{aligned}$$

Clearly,

$$fp(\mathcal{F}) \;=\; \bigcup_{n<\omega} \Sigma_n^\mu(\mathcal{F}) \;=\; \bigcup_{n<\omega} \Pi_n^\mu(\mathcal{F})$$

Remark. The choice of the letter Σ for the classes constituting μ–clones, and the letter Π for the ν–clones is motivated by the characterization of the least fixed points as the *unions* of iterations and the greatest fixed points as *intersections* (Theorem 2.2). In this respect our notation follows e.g. the standard notation for *Borel* classes. It can be noted however that the opposite choice could also be advocated, in view of the characterization given by the Knaster–Tarski Theorem (Theorem 2.1). Thus, the actual notation (also used by some other authors [12, 27]) is fixed here with some degree of arbitrariness.

Vectorial fixed points

It is often convenient to consider systems of fixed point equations; many objects, as e.g. grammars, are usually presented in that way. Readily, a system of fixed point equations over a given lattice can be considered as a single equation over a suitable product lattice; this leads us to the concept of "vectorial fixed points" that we shall analyze in this subsection more closely. It turns out that vectorial fixed points do not add any new operations to fixed point clones.

Let $h_1(y_1, \ldots y_k, z_1, \ldots, z_m), \ldots, h_k(y_1, \ldots y_k, z_1, \ldots, z_m)$ be monotonic operations on L. The product mapping $\vec{h}(\vec{y}, \vec{z}) : L^{k+m} \to L^k$ is monotonic with respect to the product ordering. We call it a *vectorial operation* on L. According to our convention, $\mu\vec{y}.\vec{h}(\vec{y}, \vec{z})$ denote the mapping from L^m to L^k that sends a vector $\vec{a}$ to the least solution of the system of equations

$$\begin{aligned} y_1 &= h_1(\vec{y}, \vec{a}), \\ &\vdots \\ y_k &= h_k(\vec{y}, \vec{a}). \end{aligned}$$

We shall show that the i-th component of that mapping, namely $\pi_k^i(\mu\vec{y}.\vec{h}(\vec{y}, \vec{z}))$, may be obtained from the operations $h_1, \ldots, h_k$, using composition and the μ-operator.

Lemma 2.6 *Let $\langle L, \leq_L \rangle$ and $\langle K, \leq_K \rangle$ be two complete lattices, let $F_1 : L \times K \to L$ and $F_2 : L \times K \to K$ be monotonic mappings, and let $\vec{F} = \langle F_1, F_2 \rangle : L \times K \to L \times K$. The two components of the least fixed point $\mu xy.\vec{F}(x, y)$ of $\vec{F}$ satisfy the following equalities:*

$$\begin{aligned} \pi_2^1(\mu xy.\vec{F}(x, y)) &= \mu x.F_1(x, \mu y.F_2(x, y)) \\ \pi_2^2(\mu xy.\vec{F}(x, y)) &= \mu y.F_2(\mu x.F_1(x, y), y). \end{aligned}$$

Proof. Let us denote by a and a' the left–hand side and right–hand side of the first equation, and b and b' those of the second equation.

We have $a = F_1(a, b)$ and $b = F_2(a, b)$, since $\langle a, b \rangle$ is a fixed point of $\vec{F}$. Hence, $\mu x.F_1(x, b) \leq a$ and $\mu y.F_2(a, y) \leq b$. By monotonicity of F_1 and F_2, $F_2(\mu x.F_1(x, b), b) \leq F_2(a, b) = b$ and $F_1(a, \mu y.F_2(a, y)) \leq F_1(a, b) = a$. It follows that $b' \leq b$ and $a' \leq a$.

On the other hand, $a' = F_1(a', \mu y.F_2(a', y))$ and $b' = F_2(\mu x.F_1(x, b'), b')$. Let $b'' = \mu y.F_2(a', y))$ and $a'' = \mu x.F_1(x, b')$. We have $a' = F_1(a', b''), b' = F_2(a'', b'), b'' = F_2(a', b'')$, $a'' = F_1(a'', b')$. It follows that $\langle a', b'' \rangle$ and $\langle a'', b' \rangle$ are fixed points of $\vec{F}$, hence, $a \leq a'$ and $b \leq b'$. □

Clearly, a similar result can be proved for greatest fixed points.

This result can be extended in the following way:

Proposition 2.7 *Let $h_i : L^{k+m} \to L$, for $1 \leq i \leq k$,be monotonic mappings. Each component of $\mu\vec{x}.\vec{h}(\vec{x}, \vec{y})$ belongs to $\mu(\{h_i \mid 1 \leq i \leq k\})$.*

Proof. Let us prove this result by induction on k. If $k = 1$, there is nothing to prove. Otherwise, let us write $F_1 = h_1$, $F_2 = \langle h_2, \ldots, h_k \rangle$, and $\vec{z} = \langle x_2, \ldots, x_k \rangle$ and let us apply Lemma 2.6 and the induction hypothesis. The first component of $\mu\vec{x}.\vec{h}(\vec{x}, \vec{y})$ is equal to $\mu x_1.h_1(x_1, \mu\vec{z}.F_2(x_1, \vec{z}, \vec{y}))$; since each component of $\mu\vec{z}.F_2(x_1, \vec{z}, \vec{y})$ is in $\mu(\{h_2, \ldots, h_k\})$, this first component is in $\mu(\{h_1\} \cup \mu(\{h_2, \ldots, h_k\})) \subseteq \mu(\{h_1, \ldots, h_k\})$. The other components are components of $\mu\vec{z}.F_2(\mu x_1.h_1(x\vec{z}, \vec{y}), \vec{z}, \vec{y})$ which are elements of $\mu(\mathrm{Comp}(\mu(\{h_1\}) \cup \{h_2, \ldots, h_k\})) \subseteq \mu(\{h_1, \ldots, h_k\})$. □

The converse of this proposition is also true and we get the following characterization of the μ-clone $\mu(\mathcal{F})$ in terms of vectorial fixed points.

Proposition 2.8 *Let $\mathcal{F}$ be a class of operations over a complete lattice, and let f be an m-ary operation over this lattice. Then f is in the μ–clone generated by $\mathcal{F}$ if and only if there exist a vectorial mapping $\vec{h} : L^{k+m} \to L^k$ in Comp$(\mathcal{F})^k$ and some i, $1 \leq i \leq k$ such that $f(\vec{y}) = \pi_k^i(\mu\vec{x}.\vec{h}(\vec{x}, \vec{y}))$.*

Proof. The "if" part of the equivalence was proved in the previous proposition. Let us prove the "only if" part by induction on the construction of f.

- if $f(y_1, \ldots, y_m)$ is a projection π_m^i, we have just to consider the equation $x = \pi_{m+1}^{i+1}(x, y_1, \ldots, y_m)$;

- if $f(y_1, \ldots, y_n)$ is in $\mathcal{F}$, we consider the equation

$$x = f(\pi^2_{m+1}(x, y_1, \ldots, y_m), \ldots, \pi^{m+1}_{m+1}(x, y_1, \ldots, y_m))$$

which is equivalent to

$$x = f(y_1, \ldots, y_m);$$

- if $f(y_1, \ldots, y_m) = g(h_1(y_1, \ldots, y_m), \ldots, h_n(y_1, \ldots, y_m))$ then by induction hypothesis, there are systems of equations
  - $\vec{x} = \vec{G}(\vec{x}, \vec{z})$, such that g is the j_0-th component of its least fixed point,
  - $\vec{x_i} = \vec{H_i}(\vec{x_i}, \vec{y})$, such that h_i is the j_i-th component of its least fixed point, for $1 \leq i \leq n$,

  with the components of $\vec{G}$ and of each $\vec{H_i}$ included in $\mathrm{Comp}(\mathcal{F})$. Then f is the j_0-th component of the least fixed point of the system

$$\begin{array}{rcl} \vec{x} & = & \vec{G}(\vec{x}, x_{1,j_1}, \ldots, x_{n,j_n}), \\ \vec{x_1} & = & \vec{H}_1(\vec{x_1}, \vec{y}), \\ & \vdots & \\ \vec{x_n} & = & \vec{H}_n(\vec{x_n}, \vec{y}). \end{array}$$

  which can be rewritten as $\vec{X} = F(\vec{X}, \vec{y})$ with $F \in \mathrm{Comp}(\{G, H_1, \ldots H_n\}) \subseteq \mathrm{Comp}(\mathcal{F})$.

- if $f(y_1, \ldots, y_m) = \mu z.g(z, y_1, \ldots, y_m)$ such that $g(z, \vec{y})$ is the j-th component of the least fixed point of $\vec{x} = \vec{G}(\vec{x}, z, \vec{y})$. Then we consider the system

$$\begin{array}{rcl} \vec{x} & = & \vec{G}(\vec{x}, z, \vec{y}) \\ z & = & x_j \end{array}$$

  where $x_j = \pi^j_k(\vec{x}, z, \vec{y})$. By Lemma 2.6, the last component of its least fixed point is precisely

$$\mu z.\pi^j_k(\mu\vec{x}.\vec{G}(\vec{x}, z, \vec{y}), z, \vec{y})$$

  which is equal to $\mu z.g(z, \vec{y})$.

$\square$

The fact stated in this proposition is sometimes called the *Bekič–Scott Principle* [39].

Example. In any complete lattice L, $\nu y_1 y_2.\mu x_1 x_2.(x_2, y_1) = (\top, \top)$, while $\mu x_1 x_2.\nu y_1 y_2.(x_2, y_1) = (\bot, \bot)$. This shows that the inequality of Proposition 2.4 is in general strict.

2.2 The calculus

We build our formalism in algebraic style. That is, the models can be in general arbitrary μ–algebras, although in the sequel we shall be mainly focused on powerset algebras. The syntax can be viewed as the usual calculus of terms built out from variables and function symbols, extended by the least and greatest fixed point operators and the logical symbol $\vee$ interpreted as lattice union. This presentation is very general and may subsume, by restriction of the class of models, several calculi considered in the literature, in particular the modal mu–calculus.

Let *Sig* be an arbitrary signature; it is considered fixed for the rest of this section.

Language

We fix a countably infinite set of variables *Var*. We assume the symbol $\vee$ is not in *Sig*. The set fpT_{Sig} of fixed point terms is defined inductively by the following clauses:

- the variables are fixed point terms,
- if $f \in Sig$ and $\tau_1, \ldots, \tau_{\rho(f)}$ are fixed point terms, so is $f(\tau_1, \ldots, \tau_{\rho(f)})$,
- if τ_1 and τ_2 are fixed point terms, so is $\tau_1 \vee \tau_2$,
- if τ is a fixed point term and x is a variable then both $\tau' = \mu x.\tau$ and $\tau'' = \nu x.\tau$ are fixed point terms.

In the sequel, we shall often refer to fixed point terms simply as terms.

An occurrence of a variable x in a term τ in a context $\eta x.\tau'$, where η is μ or ν is *bound* (by η), otherwise the occurrence is *free*. As usual, a variable is *free in a term* τ if it has a free occurrence in τ. We write $\tau \equiv \tau(x_1, \ldots, x_n)$ to indicate that the free variables of τ are among $x_1, \ldots, x_n$. A term without free variables is called *closed*.

The composition of terms, i.e. a substitution of some terms into a given term, may require first a renaming of bound variables in the underlying term, i.e. α-conversion. Any term obtained from a term τ by applying α-conversion some number of times is called a *variant* of τ. We say that a variable x is *free for a term* $\tau'(y_1, \ldots, y_m)$ *in* τ if x does not occur in scope of any ηy_i, $i = 1, \ldots, m$. Now, if $\tau(y_1, \ldots, y_m), \tau_1, \ldots, \tau_m$ are terms, the substitution $\tau[\tau_1/y_1, \ldots, \tau_m/y_m]$ is defined by first taking a variant of τ, say τ' in which each y_i is free for τ_i, and then replacing simultaneously y_i by τ_i in τ'.

Hierarchy of fixed point terms

We now define the syntactic counterparts of the classes of fixed point hierarchy. For a set of fixed point terms $F \subseteq fpT_{Sig}$, let $\mathrm{Comp}(F)$ be the least set of terms containing $F \cup \mathrm{Var}$ and closed under substitution, *viz* if $\tau(y_1, \ldots, y_m), \tau_1, \ldots, \tau_m$ are in $\mathrm{Comp}(F)$, so is $\tau[\tau_1/y_1, \ldots, \tau_m/y_m]$.

Similarly, let $\mu(F)$ be the least set of terms containing $F \cup Var$ and closed under substitution and under the μ operator, where the last means that if $\tau \in \mu(F)$ and $x \in Var$ then $\mu x.\tau \in \mu(F)$.

The set $\nu(F)$ is defined similarly.

Now let

$$\mathrm{Base}_{Sig} = \{f(x_1, \ldots, x_{\rho(f)}) : f \in Sig, x_1, \ldots, x_{\rho(f)} \in Var\} \cup \{x_1 \vee x_2\}$$

We set

$$\Sigma_0^{\mu}(Sig) = \Pi_0^{\mu}(Sig) = \mathrm{Comp}(\mathrm{Base}_{Sig})$$

$$\begin{aligned}\Sigma_{n+1}^{\mu}(Sig) &= \mu(\Pi_n^{\mu}(Sig))\\ \Pi_{n+1}^{\mu}(Sig) &= \mu(\Sigma_n^{\mu}(Sig))\end{aligned}$$

Clearly,

$$fpT_{Sig} = \bigcup_{n<\omega} \Sigma_n^{\mu}(Sig) = \bigcup_{n<\omega} \Pi_n^{\mu}(Sig)$$

Note : hierarchy of Emerson and Lei. The concept of a fixed point hierarchy considered in our paper (first introduced in [36]) is based on the ideas of D. Park [39]. A slightly different definition of a hierarchy has been proposed by Emerson and Lei [12], in context of the model checking problem for modal *mu*–calculus (c.f. Section 6.2 below). Their definition is originally based on a concept of an alternation depth of a formula which is defined "top–down". We can rephrase that definition in our setting, by inductively defining classes Σ_n^{EL} and Π_n^{EL} of fixed point terms as follows. Let $\mu_{EL}(F)$ be the closure of a set of terms F under the application of symbols in Sig and under the μ–operator; note that this class may be not closed under composition. Let $\nu_{EL}(F)$ be defined similarly. Let $\Sigma_0^{EL}(Sig) = \Pi_0^{EL}(Sig) = \mathrm{Comp}(\mathrm{Base}_{Sig})$, and let $\Sigma_{n+1}^{EL}(Sig) = \mathrm{Comp}(\mu_{EL}(\Pi_n^{EL}))$, $\Pi_{n+1}^{EL}(Sig) = \mathrm{Comp}(\nu_{EL}(\Sigma_n^{EL}))$. For example, a term $\mu x.\nu y.f(x, y, \mu z.\nu w.f(x, z, w))$, where $f \in Sig$, is in $\Sigma_2^{\mu}(Sig)$ but not in $\Sigma_2^{EL}(Sig)$. (To see that it is in $\Sigma_2^{\mu}(Sig)$, note that so are the terms $\mu z.\nu w.f(x, z, w)$, $\nu y.f(x, y, v)$ and $\nu y.f(x, y, \mu z.\nu w.f(x, z, w))$. On the other hand, our term cannot be obtained by composition of two terms in $\Sigma_2^{EL}(Sig)$ since the variable x occurs free in $\mu z.\nu w.f(x, z, w)$.) This term is actually of alternation depth 3 in the sense of Emerson and Lei [12], and, in our setting, it is not lower than in the class $\Sigma_4^{EL}(Sig)$. In general, we have only (easy) inclusions $\Sigma_n^{EL}(Sig) \subseteq \Sigma_n^{\mu}(Sig)$ and $\Pi_n^{EL}(Sig) \subseteq \Pi_n^{\mu}(Sig)$.

One argument for our choice of definition of fixed point hierarchy, especially if the expressibility issues are concerned, is provided by its compatibility with a similar classification of vectorial fixed points expressions (see Proposition 2.11 below).

Semantics

We shall interpret the fixed point terms in the μ-algebras over the signature $Sig_{\vee}$. So, let $\mathcal{A} = \langle\langle A, \{\vee_{\mathcal{A}}\} \cup \{f^{\mathcal{A}} : f \in Sig\}\rangle \leq_{\mathcal{A}}\rangle$ be a μ-algebra, where, according to our convention, $\vee_{\mathcal{A}}$ denotes the operation of the least upper bound of two elements in the lattice A. A *valuation* is any mapping from a subset $V \subseteq Var$ into A. If V is finite, say $V = \{x_1, \ldots, x_k\}$, we usually present a valuation explicitly, say, $x_1 \mapsto a_1, \ldots, x_k \mapsto a_k$, in vector notation $\vec{x} \mapsto \vec{a}$. Let τ be a fixed point term such that all the free variables of τ are among $x_1, \ldots, x_k$, in symbols $\tau \equiv \tau(x_1, \ldots, x_k)$. The *interpretation* of τ under a valuation $x_1 \mapsto a_1, \ldots, x_k \mapsto a_k$, in symbols $\tau^{\mathcal{A}}[x_1 \mapsto a_1, \ldots, x_k \mapsto a_k]$ (or $\tau^{\mathcal{A}}[\vec{x} \mapsto \vec{a}]$), is an element of A, defined by induction on the construction of τ:

- $x_i^{\mathcal{A}}[\vec{x} \mapsto \vec{a}] = a_i$,
- $f(\tau_1, \ldots, \tau_n)^{\mathcal{A}}[\vec{x} \mapsto \vec{a}] = f^{\mathcal{A}}(\tau_1^{\mathcal{A}}[\vec{x} \mapsto \vec{a}], \ldots, \tau_n^{\mathcal{A}}[\vec{x} \mapsto \vec{a}])$, for $f \in Sig$,
- $(\tau_1 \vee \tau_2)^{\mathcal{A}}[\vec{x} \mapsto \vec{a}] = \tau_1^{\mathcal{A}}[\vec{x} \mapsto \vec{a}] \vee_A \tau_2^{\mathcal{A}}[\vec{x} \mapsto \vec{a}]$,
- $(\mu y.\tau)^{\mathcal{A}}[\vec{x} \mapsto \vec{a}]$ is the least element $b \in A$, such that $b = \tau^{\mathcal{A}}[\vec{x} \mapsto \vec{a}, y \mapsto b]$,
- $(\nu y.\tau)^{\mathcal{A}}[\vec{x} \mapsto \vec{a}]$ is the greatest element $b \in A$, such that $b = \tau^{\mathcal{A}}[\vec{x} \mapsto \vec{a}, y \mapsto b]$,

Thus, any fixed point term τ, together with an ordered vector of variables $x_1, \ldots, x_k$, such that all the free variables of τ are among (but do not necessarily exhaust) $x_1, \ldots, x_k$, induces a k–ary operation on A that sends $a_1, \ldots, a_k$ on $\tau^{\mathcal{A}}[\vec{x} \mapsto \vec{a}]$. We shall denote this operation by $\tau^{\mathcal{A}}[\vec{x}]$. Note that, formally, one term τ induces an infinity of operations, depending on the vector of variables in consideration. However, when no confusion may arise, we shall sometimes write simply $\tau^{\mathcal{A}}$ instead of $\tau^{\mathcal{A}}[\vec{x}]$.

The above definitions are obviously made to allow us to prove the following.

Lemma 2.9 *Let $\tau_0 \equiv \tau_0(y_1, \ldots, y_n)$, $\tau_i \equiv \tau_i(\vec{x})$, for $i = 1, \ldots, n$ and let $\tau = \tau_0[\tau_1/y_1, \ldots, \tau_n/y_n]$. Then $\tau^{\mathcal{A}}[\vec{x}] = \tau_0^{\mathcal{A}}[\vec{y}](\tau_1^{\mathcal{A}}[\vec{x}] \ldots, \tau_n^{\mathcal{A}}[\vec{x}])$.*

The hierarchy of fixed point terms induces a hierarchy of operations definable by them in a μ–algebra $\mathcal{A}$.

Let

$$\begin{aligned}
\Sigma_n^{\mu}(\mathcal{A}) &= \{\tau^{\mathcal{A}} \,:\, \tau \in \Sigma_n^{\mu}(Sig)\} \\
\Pi_n^{\mu}(\mathcal{A}) &= \{\tau^{\mathcal{A}} \,:\, \tau \in \Pi_n^{\mu}(Sig)\} \\
fp(\mathcal{A}) &= \{\tau^{\mathcal{A}} \,:\, \tau \in fpT_{Sig}\}
\end{aligned}$$

It follows from our definitions that all the above classes are clones of operations over A; moreover $\Sigma_{n+1}^{\mu}(\mathcal{A})$ is the least family of operations containing $\Pi_n^{\mu}(\mathcal{A})$ and closed under composition and under the least fixed point operator, similarly for $\Pi_{n+1}^{\mu}(\mathcal{A})$ and the greatest fixed point.

Note : intersection. We have not included the lattice intersection $\wedge$ explicitly to the system although, in an actual μ–algebra, it can be present as one of the operations. In particular, as we have remarked in Section 1.2, the set–theoretical intersection can be retrived in a powerset algebra if the original semi–algebra contains the partial operation $\wedge$ defined there. Our system, though without explicit intersection, will be nevertheless sufficient to characterize the infinite behavior of automata over powerset algebras. This result will be proved in the next section. By considering semi–algebras with the operation $\wedge$, the characterization will comprise as a speciall case a kind of alternating automata as discussed in Section 1.3 above.

On the other hand, the presence of intersection as a logical operation would make the system intrinsically more complex. In particular, our results about initiality and hierarchy (c.f. Sections 4 and 5) depend on the absence of the intersection.

Vectorial fixed point expressions

When examining the proofs of Lemma 2.8 and Propositions 2.6 and 2.7, we can easily see that the involved transformations can be done uniformly for all *Sig*–powerset algebras, and do not depend on a particular interpretation. In particular, we note the following.

Proposition 2.10 *Let F be a set of fixed point terms, and let $\tau_1, \ldots, \tau_n \in F$, where $\tau_i \equiv \tau_i(y_1, \ldots, y_n, z_1, \ldots, z_m)$, for $i = 1, \ldots, n$. Then there exist fixed point terms $\rho_1(\vec{z}), \ldots, \rho_n(\vec{z})$ in $\mu(F)$, which represent the least solution of the system of equations*

$$\begin{array}{rcl} y_1 & = & \tau_1(\vec{y}, \vec{z}) \\ & \ldots & \\ y_n & = & \tau_n(\vec{y}, \vec{z}) \end{array}$$

That is, in any μ–algebra $\mathcal{A}$, and for any $d_1, \ldots, d_m \in A$, the vector $\rho_1^{\mathcal{A}}[\vec{z} \mapsto \vec{d}]$, $\ldots$, $\rho_n^{\mathcal{A}}[\vec{z} \mapsto \vec{d}]$ is the least fixed point *of the vectorial mapping $A^n \to A^n$,*

$$(a_1, \ldots, a_n) \mapsto (\tau_1^{\mathcal{A}}[\vec{y} \mapsto \vec{a},\, \vec{z} \mapsto \vec{d}], \ldots, \tau_n^{\mathcal{A}}[\vec{y} \mapsto \vec{a},\, \vec{z} \mapsto \vec{d}])$$

The analogous property holds for the greatest fixed points.

Consequently, if $F \subseteq \Sigma_k^\mu(\mathrm{Sig})$ then the terms $\rho_1, \ldots, \rho_n$ can be chosen in $\Pi_{k+1}^\mu(\mathrm{Sig})$, and if $F \subseteq \Pi_n^\mu(\mathrm{Sig})$ then $\rho_1, \ldots, \rho_n$ can be chosen in $\Sigma_{k+1}^\mu(\mathrm{Sig})$. $\square$

The converse to the above proposition can be also formulated, and as a consequence, one obtains a kind of characterization of operations in the classes Σ_n^μ, Π_n^μ in terms of vectorial fixed points. The proof of the following result can be obtained by repeating the argument of Proposition 2.7 several times, together with some obvious syntactic manipulations [38].

Proposition 2.11 *Let $\tau(z_1, \ldots, z_m)$ be a fixed point term in $\Sigma_n^\mu(\mathrm{Sig})$, $n \geq 1$. Then there exist $k > 0$, and a vector of terms $\sigma_1, \ldots, \sigma_k$ in $\Sigma_0^\mu(\mathrm{Sig})$,*

$$\sigma_i \equiv \sigma_i(y_{1,1}, \ldots, y_{1,k}, \ldots, y_{n,1}, \ldots, y_{n,k},\, z_1, \ldots, z_m) \text{ for } i = 1, \ldots, k,$$

such that, for any μ–algebra $\mathcal{A}$,

$$\tau^{\mathcal{A}}(\vec{z}) = \pi_n^1(\mu\vec{y_n}.\nu\vec{y_{n-1}}.\ldots.\xi\vec{y_1}.(\sigma_1^{\mathcal{A}}, \ldots, \sigma_k^{\mathcal{A}}))$$

(Here ξ stands for μ or ν depending on the parity of n.)

Moreover, the terms $\sigma_1, \ldots, \sigma_k$ can be chosen in the form $\sigma_i = \delta_1 \vee \ldots \vee \delta_{\ell_i}$, where each δ_j is either a variable or an atomic term $f(x_1, \ldots, x_{\rho(f)})$.

Conversely, if $\sigma_1, \ldots, \sigma_k$ are terms as above, then there exist fixed point terms $\tau_1, \ldots, \tau_k$ in $\Sigma_n^\mu(\mathrm{Sig})$, such that the equality

$$\tau_i^{\mathcal{A}} = \pi_n^i(\mu\vec{y_n}.\nu\vec{y_{n-1}}.\ldots.\xi\vec{y_1}.(\sigma_1^{\mathcal{A}}, \ldots, \sigma_k^{\mathcal{A}}))$$

$i = 1, \ldots, k$, holds in any μ–algebra A.

The similar result holds for the class $\Pi_n^\mu(\mathrm{Sig})$.

3 Characterization

In this section, we show that Rabin automata with variables and fixed point terms have the same expressive power over the class of powerset algebras; moreover, there is an exact correspondence between two hierarchies : the fixed point hierarchy induced by the alternations of the μ and ν operators and a hierarchy of indices of Rabin automata. This result splits naturally in two translations : from automata to fixed point terms, and from fixed point terms to automata. We give thereby an automata–theoretic characterization of the levels of the fixed point hierarchy. This main result will be completed by a discussion of related characterizations of the hierarchy, in terms of vectorial fixed point expressions and automata with the Mostowski condition, and in terms of two kinds of iteration operators : unrestricted and well–founded iteration.

Let *Sig* be an arbitrary finite signature. We fix it for the rest of this section, and henceforth we shall omit the explicit references to the signature in the notation, writing, e.g., $\mathcal{T}$ for $\mathcal{T}_{Sig}$, Σ_n^μ for $\Sigma_n^\mu(Sig)$ etc.

3.1 From automata to fixed point terms

The following lemma characterizes two basic kinds of computations of an automaton : finite and unrestricted infinite. It is a direct generalization of what is well–known in formal language theory, but we give the proof for the sake of completeness.

Lemma 3.1 *Let $A(z_1, \ldots, z_m)$ be a pre–automaton with variables and let q be a state of A. Then there exists fixed point terms ρ in Σ_1^μ and τ in Π_1^μ, such that, for any semi-algebra $\mathcal{B}$, and for any $D_1, \ldots, D_m \subseteq B$, $b \in B$,*

- $b \in \tau^{\wp\mathcal{B}}[\vec{z} \mapsto \vec{D}]$ *iff there exists a q–run of A on b w.r.t. the valuation $\vec{z} \mapsto \vec{D}$.*
- $b \in \rho^{\wp\mathcal{B}}[\vec{z} \mapsto \vec{D}]$ *iff there exists a q–run of A on b w.r.t. the valuation $\vec{z} \mapsto \vec{D}$, with* finite *domain.*

Proof. Let $A = \langle Sig, Q, \{z_1, \ldots, z_m\}, Tr \rangle$, $Q = \{x_1, \ldots x_k\}$. For each x_i, let $\bigvee Tr_i$ be the term $\alpha_1 \vee \ldots \vee \alpha_{\ell_i}$, where $x_i = \alpha_1$, ..., $x_i = \alpha_{\ell_i}$ are all the transitions in *Tr* with the head x_i; by Proviso 2 (page 14), there is at least one such transition. Consider the system of equations Eq(A) :

$$\begin{array}{rcl} x_1 & = & \bigvee Tr_1(\vec{x}, \vec{z}) \\ & \cdots & \\ x_k & = & \bigvee Tr_k(\vec{x}, \vec{z}) \end{array}$$

By Proposition 2.10, there are terms $\rho_1(\vec{z}), \ldots, \rho_n(\vec{z})$ in Σ_1^μ which, for any μ–algebra $\mathcal{A}$, represent the least solution of this system in $\mathcal{A}$, and the terms $\tau_1(\vec{z}), \ldots, \tau_n(\vec{z})$ in Π_1^μ which similarly represent its greatest solution.

Now, let $\mathcal{B}$ be a semi–algebra and let $D_1, \ldots, D_m \subseteq B$. For $i = 1, \ldots, k$, let $A_i^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$ ($A_i^{\mathcal{B}*}[\vec{z} \mapsto \vec{D}]$) denote the set of those $b \in B$ for which there exists an x_i–run (respectively, an x_i–run with finite domain) of A on b w.r.t. the valuation $\vec{z} \mapsto \vec{D}$. In order to accomplish the proof, it is enough to show that (1) $\vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$ is the greatest solution of the system Eq(A) and (2) $\vec{A}^{\mathcal{B}*}[\vec{z} \mapsto \vec{D}]$ is the least solution of this system.

Ad 1. By the Knaster–Tarski Theorem (Theorem 2.1), it is enough to show two things :

(i) $\vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}] \subseteq \bigvee \vec{Tr}^{\wp\mathcal{B}}[\vec{x} \mapsto \vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}]$,

(ii) $(\forall M_1, \ldots, M_k \subseteq B)\ \vec{M} \subseteq \bigvee \vec{Tr}^{\wp\mathcal{B}}[\vec{x} \mapsto \vec{M}, \vec{z} \mapsto \vec{D}]$ implies $\vec{M} \subseteq \vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$.

The clause (i) follows easily from the definition of a run. To show (ii), let $\vec{M}$ satisfy the hypothesis and let $b \in M_i$. An x_i–run r of A on t, w.r.t. the valuation $\vec{z} \mapsto \vec{D}$, can be constructed as follows. By hypothesis about $\vec{M}$, there is a transition $x_i = \alpha$, such that $b \in \alpha^{\wp\mathcal{B}}[\vec{x} \mapsto \vec{M}, \vec{z} \mapsto \vec{D}]$. This will be the first transition used by our run. Let $\alpha = f(y_1, \ldots, y_\ell)$. So we have $b \doteq f^{\mathcal{B}}(b_1, \ldots, b_\ell)$, for some $b_1, \ldots, b_\ell \in B$. Note that some of the y's may be among the z's and some other among the x's. But if $y_j = z_{j'}$ then b_j is in $D_{j'}$, and if $y_j = x_{j''}$ then b_j is in $M_{j''}$ and then, again by the hypothesis on $\vec{M}$, there is a transition $x_{j''} = \beta$, such that $b_j \in \beta^{\wp\mathcal{B}}[\vec{x} \mapsto \vec{M}, \vec{z} \mapsto \vec{D}]$. This transition may be also used by our run. By continuing this procedure perhaps infinitely, we obtain a desired x_i–run of A on b. The formal justification follows easily from the limit construction (Section 1.1). This completes the proof of 1.

Ad 2. Again, by the Knaster–Tarski Theorem, it is enough to show two things :

(i) $\bigvee \vec{Tr}^{\wp\mathcal{B}}[\vec{x} \mapsto \vec{A}^{\mathcal{B}*}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}] \subseteq \vec{A}^{\mathcal{B}*}[\vec{z} \mapsto \vec{D}]$,

(ii) $(\forall M_1, \ldots, M_k \subseteq B) \bigvee \vec{Tr}^{\wp\mathcal{B}}[\vec{x} \mapsto \vec{M}, \vec{z} \mapsto \vec{D}] \subseteq \vec{M}$ implies $\vec{A}^{\mathcal{B}*}[\vec{z} \mapsto \vec{D}] \subseteq \vec{M}$.

To show (i), we need to verify that, for each i, $\bigvee Tr_i[\vec{x} \mapsto \vec{A}^{\mathcal{B}*}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}] \subseteq A_i^{\mathcal{B}*}[\vec{z} \mapsto \vec{D}]$. This follows immediately from the definition of $\bigvee Tr_i$ and of a run. In order to show (ii), let $\vec{M}$ satisfy the hypothesis and let $b \in A_i^{\mathcal{B}*}[\vec{z} \mapsto \vec{D}]$ Let r be an x_i–run of A on t, w.r.t. the valuation $\vec{z} \mapsto \vec{D}$, with $\mathrm{dom}\, r$ finite. To show that $b \in M_i$, we use the Tree Induction Principle (Section 1.1). Let $S = \{w \in \mathrm{dom}\, r :$ if $r \uparrow_2 (w) = x_j$ then $r \uparrow_1 \in M_j\}$. We need to show $\epsilon \in S$. First observe that any leaf v of r is in S. Indeed, if $r \uparrow_2 (v) \in \{z_1, \ldots, z_m\}$ then the condition holds trivially, and if $r \uparrow_2 (v) = x_j$ then there must be a transition $x_j = c$ in Tr (and hence c is one of the disjuncts in $\bigvee Tr_j$), such that $r \uparrow_1 (v) \doteq c^{\mathcal{B}}$. Now the inclusion $\bigvee Tr_j^{\wp\mathcal{B}}[\vec{x} \mapsto \vec{M}, \vec{z} \mapsto \vec{D}] \subseteq M_j$ forces that $r \uparrow_1 (v) \in M_j$, as required. Next, suppose that $w \in \mathrm{dom}\, r$ is not a leaf, but all the immediate successors of w are in S. Then $\pi_2 \circ r(w)$ must be a transition in Tr, say $x_j = f(y_1, \ldots, y_{\rho(f)})$, with $\rho(f) > 0$; again, $f(y_1, \ldots, y_{\rho(f)})$ is a disjunct of $\bigvee Tr_j$. Note that some of the y_ℓ's may be among $\vec{z}$ and some others among $\vec{x}$. But, if $y_\ell = z_p$ then, by definition of a run, $r \uparrow_1 (w\ell) \in D_p$, and, if $y_\ell = x_q$ then, by hypothesis, $r \uparrow_1 (w\ell) \in M_q$. Thus, again, the inclusion $\bigvee Tr_j^{\wp\mathcal{B}}[\vec{x} \mapsto \vec{M}, \vec{z} \mapsto \vec{D}] \subseteq M_j$ implies $r \uparrow_1 (w) \in M_j$. By Tree Induction Principle, $\epsilon \in S$, and hence $b \in M_i$, as required. □

Theorem 3.2 *For any automaton with variables $A(z_1, \ldots, z_m)$, one can construct a fixed point term $\tau_A(z_1, \ldots, z_m)$, such that, for any semi–algebra $\mathcal{B}$, τ_A and A define the same operation in the powerset algebra $\wp\mathcal{B}$, that is $A^{\mathcal{B}} = \tau_A^{\wp\mathcal{B}}$. Moreover, if A is of index n, then τ_A can be chosen in Σ^μ_{2n+1}, and, additionally,*

- *if A is of index n weakened by $\emptyset$, then τ_A can be chosen in Π^μ_{2n},*

- *if A is of index n weakened by* all, *then τ_A can be chosen in Σ^{μ}_{2n},*
- *if A is of index n weakened both by $\emptyset$ and "by all", then τ_A can be chosen in Π^{μ}_{2n-1}.*

Proof. *General idea.* We proceed by induction on the length of the accepting condition and distinguish two cases. If the condition has form $\{(L_1, U_1), \ldots, (L_{n-1}, U_{n-1}), (\emptyset, U_n)\}$, we consider an automaton in which the states of U_n are moved into variables, by which the index of automaton decreases. Then the original automaton is characterized by application of the greatest fixed point operator to the terms which by induction hypothesis characterize the modified automaton. If no pair in the accepting condition has the form $(\emptyset, U)$, situation is a bit more difficult. We consider n auxiliary automata in which the states of the sets $L_1, \ldots, L_n$, are moved to variables, respectively. Now the least fixed point operator is applied. Note that the negative (L) and positive (U) constraints of the Rabin condition are captured by the least and greatest fixed point operators, respectively.

In order to make the concept of "moving states into variables" more precise, the following two operations on automata will be used. The first consists in converting some states into variables. Let $A = \langle Sig, Q, V, q_0, Tr, Acc\rangle$ be an automaton with variables and let X be a subset of Q such that $q_0 \notin X$. We define an automaton[7] $A_{free(X)} = \langle Sig, Q - X, V \cup X, q_0, Tr', Acc'\rangle$, where a transition $y = f(x_1, \ldots, x_k)$ is in Tr' iff it is in Tr and $y \in Q - X$ (while the x_i's may be in $Q \cup V$), and the acceptance condition is restricted to $Q - X$, that is, $Acc' = \{(L - X, U - X) : (L, U) \in Acc\}$.

The second operation exchanges initial states, but for technical reasons, a new copy of the new initial state is added. For A as above and $x \in Q$, let $\hat{x}$ be a symbol not in $Q \cup V$. We define an automaton $A_{start(x)} = \langle Sig, Q \cup \{\hat{x}\}, V, \hat{x}, Tr'', Acc\rangle$, where $Tr'' = Tr \cup \{\hat{x} = f(x_1, \ldots, x_k) : x = f(x_1, \ldots, x_k) \text{ is in } Tr\}$. Note that introduction of $\hat{x}$ makes possible a composition of both operations, $A_{start(x)free(X)}$, also when $x \in X$. However, semantically, the automaton $A_{start(x)}$ is clearly equivalent to the automaton A with the initial state replaced by x; in particular $L(A_{start(q_0)})$ coincides with $L(A)$.

Now let $A = \langle Sig, Q, \{z_1, \ldots, z_m\}, q_0, Tr, \{(L_1, U_1), \ldots, (L_n, U_n)\}\rangle$. We are going to prove the claim of Theorem by induction on the index of the automaton A. If $n = 0$, no infinite path is allowed in any accepting run of A. Then, the result follows from Lemma 3.1 (remember that the index 0 cannot be weakened). Another easy case that we shall consider separately, is when the acceptance condition contains the pair $\{(\emptyset, Q)\}$; in this case *any* run of A is accepting. Then, again by Lemma 3.1, we can find a fixed point term in Π^{μ}_1, equivalent to A.

Now suppose that $n > 0$ and the claim holds for all automata with an index less than n. The proof proceeds in two steps.

Step 1. The acceptance condition has the form $Acc = \{(L_1, U_1), \ldots, (L_{n-1}, U_{n-1}), (\emptyset, U_n)\}$. We shall prove that one can construct an equivalent fixed point term in Π^{μ}_{2n} and

[7] The notation is motivated by an analogy between the rôle of states in an automaton and that of the bound variables in a fixed point term. Thus, the operation $A_{free(X)}$ consists in "giving freedom" to variables in X.

if moreover, for some i, $U_i \cup L_i = Q$, then the fixed point term can be chosen in Π^{μ}_{2n-1}. Both cases will be considered simultaneously.

If $U_n = \emptyset$ then clearly we can delete the pair $(\emptyset, \emptyset)$ from the acceptance condition without changing the semantics of A, therefore, by the induction hypothesis, the automaton is equivalent to a fixed point term in $\Sigma^{\mu}_{2n-2} \subseteq \Pi^{\mu}_{2n-1}$. If $U_n = Q$ we have the case that has been treated above. So, we may assume $\emptyset \neq U_n \neq Q$. We can also assume that the initial state q_0 is not in U; otherwise we would take into consideration a semantically equivalent automaton $A_{start(q_0)}$.

To simplify notation, let $U_n = U$. Let $U = \{y_1, \ldots, y_k\}$. We shall consider the automata $A_{free(U)}$, and $A_{start(y_i)free(U)}$, for $i = 1, \ldots, y_k$ (where the last is a simplified notation for $(A_{start(y_i)})_{free(U)}$). The acceptance condition of each of these automata is $\{(L_1, U_1), \ldots, (L_{n-1}, U_{n-1}), (\emptyset, \emptyset)\}$. After deleting the useless pair $(\emptyset, \emptyset)$, we can consider all these automata being of index $n-1$. Then, by the induction hypothesis, there exist fixed point terms in Σ^{μ}_{2n-1}, $\tau_0(y_1, \ldots, y_k, z_1, \ldots, z_m)$ and $\tau_1(\vec{y}, \vec{z}), \ldots, \tau_k(\vec{y}, \vec{z})$, such that, for any semi–algebra $\mathcal{B}$, $A^{\mathcal{B}}_{free(U)} = \tau^{\wp\mathcal{B}}$ and $A^{\mathcal{B}}_{start(y_i)free(U)} = \tau_i^{\wp\mathcal{B}}$, for $i = 1, \ldots, k$.

If additionally the stronger hypothesis on the acceptance condition holds, namely that $L_i \cup U_i = Q$, for some i (but not for $i = n$; that case has been already considered), then these fixed point terms can be chosen in Σ^{μ}_{2n-2}.

Consider a system of equations

$$\begin{aligned} y_1 &= \tau_1 \\ &\cdots \\ y_k &= \tau_k \end{aligned}$$

By Proposition 2.11, there is a vector of fixed point terms in Π^{μ}_{2n}, $\rho_1(\vec{z}), \ldots, \rho_k(\vec{z})$ which, for any semi–algebra $\mathcal{B}$, represent the greatest solution of this system in $\wp\mathcal{B}$. In the case when the stronger hypothesis on the acceptance condition holds, the terms $\rho_1(\vec{z}), \ldots, \rho_k(\vec{z})$ can be chosen in Π^{μ}_{2n-1}.

We claim that, for any semi–algebra $\mathcal{B}$,

1. $A^{\mathcal{B}}_{start(y_i)} = \rho_i(\vec{z})^{\wp\mathcal{B}}$, for $i = 1, \ldots, k$,
2. $A^{\mathcal{B}} = \tau_0[\rho_1/y_1, \ldots, \rho_k/y_k]^{\wp\mathcal{B}}$.

Note that the term of (2) is in Π^{μ}_{2n}, and if, additionally, the index of A is weakened by $\emptyset$, it is in Π^{μ}_{2n-1}. Hence, by proving (2), we shall accomplish the Step 1.

We first show that (2) follows from (1).

Let us fix some $D_1, \ldots, D_m \subseteq B$, we have to show $A^{\mathcal{B}}[\vec{z} \mapsto \vec{D}] = \tau_0[\vec{\rho}/\vec{y}]^{\wp\mathcal{B}}[\vec{z} \mapsto \vec{D}]$.

"$\subseteq$" Let r be an accepting run of A on $b \in B$ w.r.t. the valuation $\vec{z} \mapsto \vec{D}$. By choice of τ_0, it is enough to construct an accepting run of $A_{free(U)}$ on b, w.r.t. the valuation $\vec{y} \mapsto \vec{\rho}^{\wp\mathcal{B}}[\vec{z} \mapsto \vec{D}]$, $\vec{z} \mapsto \vec{D}$. Let E be the set of these nodes in $\operatorname{dom} r$, where some state from U occurs for the first time (remember that the initial state q_0 is not in U), that is $E = \{w \in \operatorname{dom} r : r\uparrow_2(w) \in U \text{ and } (\forall v < w)\, r\uparrow_2(v) \notin U\}$. Clearly, E is an antichain. We define a tree r_0 by cutting the branches of r in the nodes of E; more precisely, we define it by substitution (c.f. Section 1.1) $r_0 = r[trunc]$, where the mapping $trunc$ sends each $v \in E$ on a tree $r_v : \{\epsilon\} \to B \times Tr$ defined by $r_v(\epsilon) = \langle r\uparrow_1(v), r\uparrow_2(v)\rangle$.

We claim that r_0 is a desired run of $A_{free(U)}$ on t. Indeed, $r_0 \uparrow_2 (v) \in U$ is possible only if $v \in E$, and, for any $v \in E$, say $r(v) = y_j$, the subtree $r.v$ of r constitutes an accepting y_j–run of A on $r \uparrow_1 (v)$, and hence clearly $r \uparrow_1 (v) \in A^{\mathcal{B}}_{start(y_j)}[\vec{z} \mapsto \vec{D}]$. Then, assuming (1), we have $r \uparrow_1 (v) \in \rho_j^{\wp\mathcal{B}}[\vec{z} \mapsto \vec{D}]$. Satisfaction of the remaining properties of an accepting run is straightforward.

"$\supseteq$". In this case, by choice of τ_0, we have an accepting run of $A_{free(U)}$ on b, w.r.t. the valuation $\vec{y} \mapsto \vec{\rho}^{\wp\mathcal{B}}[\vec{z} \mapsto \vec{D}]$, $\vec{z} \mapsto \vec{D}$, say r_0. For each $w \in \operatorname{dom} r_0$ such that $r_0 \uparrow_2 (w) \in U$, say $r_0 \uparrow_2 (w) = y_j$, we have $r_0 \uparrow_1 (w) \in \rho_j^{\wp\mathcal{B}}[\vec{z} \mapsto \vec{D}]$, and, assuming (1), the last equals to $A^{\mathcal{B}}_{start(y_j)}[\vec{z} \mapsto \vec{D}]$. Hence we have also an accepting y_j–run of A on $r_0 \uparrow_1 (w)$. Now, by substituting a suitable run of A into each node w of r_0 such that $r_0 \uparrow_2 (w) \in U$, we obtain the desired run of A on b.

It remains to show (1). Let us fix a valuation $z_1 \mapsto D_1, \ldots, z_m \mapsto D_m$, where $D_1, \ldots, D_m \subseteq B$. Let us abbreviate,

$$\vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}] =_{def} (A^{\mathcal{B}}_{start(y_1)}[\vec{z} \mapsto \vec{D}], \ldots, A^{\mathcal{B}}_{start(y_k)}[\vec{z} \mapsto \vec{D}])$$

We need to verify that $\vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$ is indeed the greatest solution of the system in consideration. By the Knaster–Tarski Theorem (2.1), it is enough to show two things :

(i) $(\forall M_1, \ldots, M_k \subseteq B)\ \vec{M} \subseteq \vec{\tau}^{\wp\mathcal{B}}[\vec{y} \mapsto \vec{M}, \vec{z} \mapsto \vec{D}]$ implies $\vec{M} \subseteq \vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$.

(ii) $\vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}] \subseteq \vec{\tau}^{\wp\mathcal{B}}[\vec{y} \mapsto \vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}]$.

Ad (i). Suppose $\vec{M}$ satisfies the hypothesis and let $b \in M_i$, for some i. We have to show $b \in A^{\mathcal{B}}_{start(y_i)}[\vec{z} \mapsto \vec{D}]$. For this, it is enough to construct a successful y_i–run of A on t w.r.t. the valuation $\vec{z} \mapsto \vec{D}$. We obtain such a run by a limit construction. By hypothesis about $\vec{M}$, $b \in \tau_i^{\wp\mathcal{B}}[\vec{y} \mapsto \vec{M}, \vec{z} \mapsto \vec{D}]$. Hence, by hypothesis about τ_i, there is a successful run of $A_{start(y_i)free(U)}$ on b, w.r.t. the valuation $\vec{y} \mapsto \vec{M}, \vec{z} \mapsto \vec{D}$. We fix one such run, say r_0. (Recall that $r_0 \uparrow_2 (\epsilon) = \hat{y}_i$.) Observe that any node w of r_0, such that $r_0 \uparrow_2 (w) \in U$, must be a leaf, and, moreover, if $r_0(w) = y_j$ then $r \uparrow_1 (w) \in M_j$; hence, again, there is a successful run of $A_{start(y_j)free(U)}$ on $r \uparrow_1 (w)$, w.r.t. the valuation $\vec{y} \mapsto \vec{M}, \vec{z} \mapsto \vec{D}$. We define r_1 by substituting such a run in each leaf w of r_0 such that $r_0 \uparrow_2 (w) \in U$. It follows from the construction that $\operatorname{dom}(r_0) \subseteq \operatorname{dom}(r_1) \subseteq \operatorname{dom}(t)$, and, again, any node w of r_1 such that $r_1 \uparrow_2 (w) \in U$, must be a leaf of r_1, and, moreover, if $r_1 \uparrow_2 (w) = y_\ell$, we have $r \uparrow_1 (w) \in M_\ell$. So, we can repeat the construction again and again, defining a sequence of trees r_n in the obvious way. Clearly, this sequence satisfies the convergence requirement and then has a limit $\lim r_n$. Note first that, by construction, no finite path of $\lim r_n$ may end with a value y_ℓ. Then observe that any infinite path of $\lim r_n$ is either a completion of some path of some accepting run of the automaton $A_{start(y_j)free(U)}$, for some j, or it contains an infinite number of occurrences of some "initial transition" of some $A_{start(y_j)free(U)}$, $\hat{y}_j = \gamma$. Let r be a modification of $\lim r_n$ obtained by "removing hats" from the $\hat{y}_\ell$'s in all values of $\lim r_n$. We claim that r is an accepting y_i–run of A on b. Indeed, it follows from the remark above that all infinite paths of r are accepting. The remaining conditions of a successful run are obviously satisfied.

Ad (ii). The proof is very similar to the proof of the inclusion "$\subseteq$"of the claim (2) above and is omitted. The *Step 1* of the induction is now completed.

Step 2. We consider an automaton $A(z_1, \ldots, z_m)$ as above but this time we do not assume that the acceptance condition $((L_1, U_1), \ldots, (L_{n-1}, U_{n-1}), (L_n, U_n))$ is weakened by $\emptyset$. We shall construct an equivalent fixed point term of level Σ^{μ}_{2n+1} and if, additionally, for some i, $L_i \cup U_i = Q$, this fixed point term will of actually smaller level Σ^{μ}_{2n}.

Let $L = L_1 \cup \ldots \cup L_n$ and let $L = \{x_1, \ldots, x_p\}$. We may assume without loss of generality that $q_0 \notin L$. Consider first the automaton $A_{free(L)}$. The acceptance condition of this automaton is $\{(\emptyset, U_i - L) : i = 1, \ldots, n\}$, and it should be clear that without changing the semantics it can be collapsed to just one pair $(\emptyset, U)$, with $U = \bigcup_{i=1,\ldots,n} U_i - L$. Therefore, by the Step 1, there exists a fixed point term τ_0 in Π^{μ}_2 (hence in Π^{μ}_{2n}), $\tau_0 \equiv \tau_0(\vec{x}, \vec{z})$, such that $A^{\mathcal{B}}_{free(L)} = \tau_0^{\wp\mathcal{B}}$, for any semi–algebra $\mathcal{B}$. If, moreover, for some i, $L_i \cup U_i = Q$, then clearly U equals to the set of all states of the automaton $A_{free(L)}$; in this case the term τ_0 can be chosen in Π^{μ}_1.

Next, for each $x \in L$ and each $i \in \{1, \ldots, n\}$, consider the automaton $A_{start(x)free(L_i)}$. The acceptance condition of this automaton is in the form that has been considered in the *Step 1* (since the pair (L_i, U_i) is replaced by $(\emptyset, U_i - L_i)$). Then, we already know how to construct a fixed point term in Π^{μ}_{2n}, say $\tau_{x,i}$, equivalent to $A_{start(x)free(L_i)}$. If, additionally, for some $j \in \{1, \ldots, n\}$, $L_j \cup U_j = Q$, there are two cases to consider. If $j \neq i$, then, by induction hypothesis, the term $\tau_{x,j}$ can be chosen in Π^{μ}_{2n-1}, and if, for i itself, $L_i \cup U_i = Q$ then, similarly as above, this term can be chosen even in Π^{μ}_1.

Let, for each $x_\ell \in L$,

$$\tau_\ell \quad = \quad \tau_{x_\ell,1} \vee \ldots \vee \tau_{x_\ell,n}$$

(note that the free variables of τ_ℓ are among $x_1, \ldots, x_p$, $z_1, \ldots, z_m$).

Consider a system of equations

$$\begin{array}{rcl} x_1 & = & \tau_1 \\ . & . & . \\ x_p & = & \tau_p \end{array}$$

By Proposition 2.11, there is a vector of fixed point terms in Σ^{μ}_{2n+1}, $\rho_1(\vec{z}), \ldots, \rho_p(\vec{z})$, which, for any semi–algebra $\mathcal{B}$, represent the least solution of this system in $\wp\mathcal{B}$. (In the case of the stronger hypothesis on *Acc*, these terms can be chosen in Σ^{μ}_{2n}.)

We claim that, for any semi–algebra $\mathcal{B}$,

1. $A^{\mathcal{B}}_{start(x_i)} = \rho_i(\vec{z})^{\wp\mathcal{B}}$, for $i = 1, \ldots, p$,

2. $A^{\mathcal{B}} = \tau_0[\rho_1/x_1, \ldots, \rho_p/x_p]^{\wp\mathcal{B}}$.

Note that the last term is in Σ^{μ}_{2n+1}, and if, for some j, $L_j \cup U_j = Q$, this term can be chosen in Σ^{μ}_{2n}. Therefore, by proving (2), we shall accomplish the Step 2. The proof that (2) follows from (1) is similar to the proof of an analogous statement in the *Step 1*, and will be omitted.

We are now going to prove (1).

Again, we fix a valuation $z_1 \mapsto D_1, \ldots, z_m \mapsto D_m$, where $D_1, \ldots, D_m \subseteq B$, and abbreviate,

$$\vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}] = (A^{\mathcal{B}}_{start(x_1)}[\vec{z} \mapsto \vec{D}], \ldots, A^{\mathcal{B}}_{start(x_p)}[\vec{z} \mapsto \vec{D}])$$

We need to verify that $\vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$ is indeed the least solution of the system in consideration. By the Knaster–Tarski Theorem (Theorem 2.1), it is enough to show two things :

(i) $(\forall M_1, \ldots, M_k \subseteq B)\ \vec{\tau}^{\wp\mathcal{B}}[\vec{y} \mapsto \vec{M}, \vec{z} \mapsto \vec{D}] \subseteq \vec{M}$ implies $\vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}] \subseteq \vec{M}$

(ii) $\vec{\tau}^{\wp\mathcal{B}}[\vec{y} \mapsto \vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}] \subseteq \vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$

Ad (i). Suppose $\vec{M}$ satisfies the condition above and $b \in A^{\mathcal{B}}_{start(x_i)}[\vec{z} \mapsto \vec{D}]$. So we have an accepting x_i–run of $A_{start(x_i)}$ on b, w.r.t. the valuation $[\vec{z} \mapsto \vec{D}]$; fix such a run, say r.

We fix an infinite sequence of integers $a_q \in \{1, \ldots, n\}$ that repeats each $\ell \in \{1, \ldots, n\}$ infinitely many times; for instance let $a_q = (q \bmod n) + 1$. We define a sequence of subsets of dom$r$, E_q, inductively as follows :

$$\begin{aligned} E_0 &= \{\epsilon\} \\ E_{q+1} &= \{v : r \uparrow_2 (v) \in L_{a_{q+1}} \wedge (\exists w \in E_q)\, w < v \wedge (\forall u, w < u < v)\, r \uparrow_2 (u) \notin L_{a_{q+1}}\} \end{aligned}$$

Notice that E_q need not be finite. Let $E = \bigcup_{q<\omega} E_q$. The key observation is that no infinite path of r can meet the set E infinitely often since otherwise such a path would not be accepting, as each set L_j would be visited infinitely many times. We want to show that, for each $w \in E$, if $r \uparrow_2 (w) = x_j$ then $r \uparrow_1 (w) \in M_j$, in particular $b \in M_i$. To this end, it is convenient to organize the set E into a well–founded tree and use the Tree Induction Principle. The idea of such a tree should be apparent : just delete all the nodes of r *not* in E, keeping the seniority relation of r.

Formally, we define a (non–valued) tree T_E as a set of sequences over E, so that the nodes of T_E are sequences of nodes of r (to avoid the ambiguity of notation, we shall write the nodes of r in parentheses $\langle\rangle$). Let

$$T_E = \{\epsilon\} \cup \{\langle w_1 \rangle \ldots \langle w_\ell \rangle : w_1 \in E_1 \wedge w_2 \in E_2 \wedge \ldots \wedge w_\ell \in E_\ell \wedge w_1 < \ldots < w_\ell\}$$

Clearly, T_E is a tree. By remark above, it is well–founded, although it need not be finite due to infinite branching.

Let, for $\alpha \in T_E$,

$$\begin{aligned} \text{last}(\alpha) &= w_\ell \quad \text{if } \alpha = \langle w_1 \rangle \ldots \langle w_\ell \rangle \\ &= \epsilon \quad \text{if } \alpha = \epsilon \end{aligned}$$

We claim that, for each $\alpha \in T_E$, if $r \uparrow_2 (\text{last}(\alpha)) = x_j$ then $r \uparrow_1 (\text{last}(\alpha)) \in M_j$, in particular $b \in M_i$ as required.

By Tree Induction Principle, it is enough to verify that if all immediate successors of α satisfy the condition, so does α. Let last$(\alpha) = w$, $w \in E_\ell$ for some $\ell \geq 0$, and $r \uparrow_2 (w) = x_j$. Let $r \uparrow_1 (w) = b'$. The subtree $r.w$ of r constitutes an accepting x_j–run of the automaton A on the element b'. Let $F = \{v \in \text{dom}\, r.w : wv \in E_{\ell+1}\}$. Clearly, F is an antichain and, for each $v \in F$, $\alpha\langle wv \rangle$ is an immediate successor of α in T_E. Then, by the tree induction hypothesis, if $r.w \uparrow_2 (v) = r \uparrow_2 (wv) = x_k$ then $r.w \uparrow_2 (v) = r \uparrow_2 (wv) \in M_k$. Using this observation, we can transform the x_j–run of A on b', $r.w$, to an accepting run of the automaton $A_{start(x_j) free(L_{\ell+1})}$ on the same element b', under the valuation $\vec{x} \mapsto \vec{M}$, $\vec{z} \mapsto \vec{D}$. Then, by definition of the terms τ, we get $b' \in \tau_j^{\wp\mathcal{B}}[\vec{x} \mapsto \vec{M}, \vec{z} \mapsto \vec{D}]$. Hence, by the hypothesis on $\vec{M}$, $b' \in M_j$, as required. This remark completes the proof of (i).

Ad (ii). Let $b \in \tau_i^{\wp\mathcal{B}}[\vec{y} \mapsto \vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}]$. Then, for some $j \in \{1, \ldots, n\}$, $b \in \tau_{x_i,j}^{\wp\mathcal{B}}[\vec{y} \mapsto \vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}]$. Hence, there exists an accepting run of the automaton $A_{start(x_i) free(L_j)}$ on b, under the valuation $\vec{y} \mapsto \vec{A}^{\mathcal{B}}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}$, say r_0. Note that, for each $w \in \mathrm{dom} r_0$, such that $r_0 \uparrow_2 (w) \in L_j$, say $r_0 \uparrow_2 (w) = x_\ell$, we have an accepting x_ℓ–run of the automaton A on $r \uparrow_1 (w)$. Then, by a suitable substitution, we can easily obtain an accepting run of $A_{start(x_i)}$ on b, as required.

Proof of the theorem is now completed. $\square$

Note on complexity

The above proof of Theorem 3.2 can be implemented as a recursive algorithm which takes as an input a Rabin automaton and produces an equivalent fixed point term. We can therefore estimate the size of this term w.r.t. the size of the automaton.

Let a Rabin automaton A have m states and n pairs in the accepting condition. Note that n can be exponentially higher than m. Also note that the transition table of A has the size $m^{0(1)}$, where the constant depends on the signature. We first observe that, by the proof of Proposition 2.7, a term expressing a component of the least (or greatest) solution of a system $x_i = \tau_i$, $i = 1, \ldots, p$, is of the size $O((|\tau_1| + \ldots + |\tau_p|)^p)$. In order to perform *Step 1* in our recursive algorithm, we first call recursively the procedure for the automata $A_{free(U)}$ and $A_{start(y_i) free(U)}$, $i = 1, \ldots, k$, by which the terms τ, $\tau_1, \ldots, \tau_k$ are computed (page 32). Then the system $y_1 = \tau_1, \ldots, y_k = \tau_k$ is solved and its solution, $\rho_1, \ldots, \rho_k$, is substituted to τ, in order to get the result. It should be noted that all the constructions performed on automata by our algorithm do not increase the number of states. Therefore, in each recursive performance of *Step 1* in the computation initially started with A, we have $k \leq m$. Thus the term resulting from *Step 1* has the size $O(|\tau| \cdot (|\tau_1| + \ldots + |\tau_k|)^m)$, which can be further estimated by $O(m^m \cdot (\max\{|\tau|, |\tau_1|, \ldots, |\tau_k|\})^{m+1})$.

The *Step 2* (page 34) is even more costly, as we have to make $i \cdot O(m)$ recursive calls rather than $O(m)$ calls, where i is the number of pairs of the actually considered automaton (so this is n for the original automaton and decreases by 1 after each call of *Step 2*). If max is the maximum size of the terms computed by the recursive calls of *Step 2*, then the size of the system of equations to be solved is bounded by $i \cdot m \cdot max$. Considerations as above give us an estimation $O((m \cdot i)^m max^{m+1})$ on size of the term resulting from *Step 2*. Note that at the bottom of the recursion, the algorithm has to consider an automaton with an empty or trivial acceptance condition, and with the set of transitions being a subset of the original *Tr*. Therefore, the above considerations lead to an estimation of the size of the eventually computed term by $2^{c \cdot m^{2n}} \cdot (n!)^m$, for some constant c.

It seems, however, that the result may be better if we work with *vectorial* fixed point expressions (c.f. Proposition 2.11) instead of fixed point terms. Indeed, suppose that at each performance of *Step 1* or *Step 2*, our algorithm computes a vectorial expression equivalent to the actually considered automaton. In that case, the size of the expression produced in *Step 1* can be estimated by the maximal size of the expressions computed by recursive calls multiplied by m, whereas, in *Step 2*, the multiplication factor will be $m \cdot i$, by the reasons explained above. Therefore, the final vectorial expression equivalent to the original automaton will be of the size $m^{O(n)} \cdot n!$.

3.2 From fixed point terms to automata

Translation of fixed point expressions to automata is very natural; we recall here our construction of [36] for the sake of completeness, and in order to close up the connection between the fixed point hierarchy and the hierarchy of Rabin indices of automata indicated by Theorem 3.2. An analogous translation for the modal mu–calculus has been made by Streett and Emerson [46], and is also discussed in [13] and [21].

For technical reasons, it is convenient to extend slightly our concept of automata with variables, by adding one more feature : *initial variables*. Such an extension is necessary if we wish to give an automata characterization to the terms in which a free variable may occur outside of the scope of any functional symbol from the signature, as e.g., $z_1 \vee z_2$. In the remainder of this section, we shall consider automata with variables of the form $A = \langle Sig, Q, V, V_I, q_0, Tr, Acc \rangle$ where $V_I \subseteq V$ is a set of *initial variables* (it may be empty). The semantics of automata is extended as follows. Let $V = \{z_1, \ldots, z_m\}$, let $\mathcal{B}$ be a semi–algebra over Sig and let $D_1, \ldots, D_m \subseteq B$. All the runs formerly accepting, continue to be accepting. Additionally, if an element b of B is in D_i and $z_i \in V_I$ is an initial variable, then a one element tree r with $\operatorname{dom} r = \{\epsilon\}$ and $r(\epsilon) = \langle b, z_i \rangle$ is considered as an accepting q_0–run of A on b, with respect to the valuation $\vec{z} \mapsto \vec{D}$. We shall call such a run *lazy*. Note that if A' is an automaton obtained from A by forgetting the issue of initial variables (but keeping the set of variables V intact) then the operation defined by A satisfies

$$A^{\mathcal{B}}[\vec{z} \mapsto \vec{D}] \;=\; A'^{\mathcal{B}}[\vec{z} \mapsto \vec{D}] \cup \bigcup_{z_i \in V_I} D_i$$

Theorem 3.3 *For any fixed point term $\tau(z_1, \ldots, z_m)$, one can construct an automaton with variables $A_\tau(z_1, \ldots, z_m)$, such that, for any semi–algebra $\mathcal{B}$, τ and A_τ define the same operation in the powerset algebra $\wp\mathcal{B}$, that is,*

$$\tau^{\wp\mathcal{B}} \;=\; A_\tau^{\mathcal{B}}[\vec{z}]$$

Moreover, the automaton A_τ can be chosen with the Rabin acceptance condition in the chain form, and,

1. *if τ is in Σ^{μ}_{2n+1} then A_τ can be chosen with index n,*
2. *if τ is in Σ^{μ}_{2n}, $n > 0$, then A_τ can be chosen with index n weakened "by all",*
3. *if τ is in Π^{μ}_{2n+1}, then A_τ can be chosen with index $n+1$ weakened by $\emptyset$ and "by all",*

4. *if τ is in Π^{μ}_{2n}, $n > 0$, then A_τ can be chosen with index n weakened by $\emptyset$,*

5. *if τ is in $\Sigma^{\mu}_0 = \Pi^{\mu}_0$, then A_τ can be chosen with empty acceptance condition.*

We start with a series of lemmas which show that the operations definable by automata are closed under the fixed point operators and composition. Let $A(x, z_1, \dots, z_m)$ be an automaton. Recall that, by our notational convention, $\mu x.A^{\mathcal{B}}[x, \vec{z}]$ denotes the mapping that sends a tuple $\vec{D}$ of subsets of B onto the least fixed point of the equation $M = A^{\mathcal{B}}[x \mapsto M, \vec{z} \mapsto \vec{D}]$, where M ranges over $\wp B$.

Lemma 3.4 *Let $A(x, \vec{z})$ be an automaton of index n, $n \geq 0$. Then there exists an automaton $A'[\vec{z}]$ of the same index such that, for any semi–algebra $\mathcal{B}$, $A'^{\mathcal{B}}[\vec{z}] = \mu x.A^{\mathcal{B}}[x, \vec{z}]$. Moreover, if the index of A is weakened "by all" then so is the index of A'* (warning : the similar does not apply to the weakening by $\emptyset$)*, and, if A is in chain form, so is A'.*

Proof of the lemma. This is in fact a slight refinement of the well–known construction that produces an finite automaton recognizing language L^* from an automaton recognizing language L.

Let $A = \langle Sig, Q, \{x, z_1, \dots, z_m\}, V_I, q_0, Tr, Acc\rangle$ with $Acc = \{(L_1, U_1), \dots, (L_n, U_n)\}$ (if $n = 0$, Acc is empty). We define an automaton $A' = \langle Sig, Q \cup \{x\}, \{z_1, \dots, z_m\}, V_I \setminus \{x\}, x, Tr', Acc'\rangle$, where

- $Tr' = Tr \cup \{x = f(y_1, \dots, y_{\rho(f)})$: if $q_0 = f(y_1, \dots, y_{\rho(f)})$ is a transition in $Tr\}$
- $Acc' = \{(L_1 \cup \{x\}, U_1), \dots, (L_n \cup \{x\}, U_n)\}$
  (Note that if $Acc = \emptyset$ then $Acc' = \emptyset$, too.)

Let $\mathcal{B}$ be an arbitrary semi–algebra and let $\vec{z} \mapsto \vec{D}$ be an arbitrary valuation, where $D_1, \dots, D_m \subseteq B$. It remains to verify that $A'^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$ is indeed the least fixed point of the mapping $M \mapsto A^{\mathcal{B}}[x \mapsto M, \vec{z} \mapsto \vec{D}]$, where M ranges over $\wp B$.

By the Knaster–Tarski Theorem, it is enough to show two things :

(i) $(\forall M \subseteq B)\ A^{\mathcal{B}}[x \mapsto M, \vec{z} \mapsto \vec{D}] \subseteq M$ implies $A'^{\mathcal{B}}[\vec{z} \mapsto \vec{D}] \subseteq M$,

(ii) $A^{\mathcal{B}}[x \mapsto A'^{\mathcal{B}}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}] \subseteq A'^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$.

The argument is similar to the one given for an analogous statement in the proof of Theorem 3.2.

Ad (i). Suppose $\vec{M}$ satisfies the hypothesis, and let $b \in A'^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$. Let r be an accepting run of A' on t, with respect to the valuation $\vec{z} \mapsto \vec{D}$. If r is a lazy run with $r(\epsilon) = z_j$ for some $z_j \in V_I \setminus \{x\}$ then $b \in D_j$ and hence $b \in A^{\mathcal{B}}[x \mapsto M, \vec{z} \mapsto \vec{D}]$, from which the assertion $b \in M$ follows by hypothesis on M.

Then suppose r is not lazy and so $\pi_2 \circ r(\epsilon) = x$. Let $E = \{w \in \operatorname{dom} r : \pi_2 \circ r(w) = x\}$. Observe that no infinite path of r may intersect the set E infinitely often, otherwise a path would not be accepting. We shall organize the set E into a well–founded tree and use the Tree Induction Principle, similarly as in *Step 2* of the proof of Theorem 3.2.

Let us write $v \ll w$ to mean "$v < w$ and, for all u, such that $v < u < w$, $u \notin E$".

Let

$$T_E = \{\epsilon\} \cup \{(w_1)\ldots(w_\ell) : w_1, \ldots, \wedge w_\ell \in E \wedge \epsilon \ll w_1 \ll \ldots \ll w_\ell\}$$

Clearly, T_E is a well–founded tree. Let, for $\alpha \in T_E$,

$$\begin{aligned} \mathrm{last}(\alpha) &= w_\ell \quad \text{if } \alpha = (w_1)\ldots(w_\ell) \\ &= \epsilon \quad \text{if } \alpha = \epsilon \end{aligned}$$

We claim that, for each $\alpha \in T_E$, if $\pi_2 \circ r(\mathrm{last}(\alpha)) = x$ then $t.\mathrm{last}(\alpha) \in M$, in particular $t \in M$ as required. The argument is similar to that used in the proof of an analogous statement in *Step 2* of the proof of Theorem 3.2 and will be omitted.

Ad (ii). Suppose $b \in A^{\mathcal{B}}[x \mapsto A'^{\mathcal{B}}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}]$ and let r be an accepting run of A on b, with the valuation $x \mapsto A'^{\mathcal{B}}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}$. We consider three cases :

1. r is a lazy accepting run with $r\uparrow_2(\epsilon) = z_j$ for some $z_j \in V_I$ and $z_j \neq x$. Then r is also an accepting run of A'.
2. It happens that $x \in V_I$, and r is a lazy accepting run with $r\uparrow_2(\epsilon) = x$. Thus, by definition of a lazy run, $b \in A'^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$ as required.
3. r is an accepting run with $r\uparrow_2(\epsilon) = q_0$. Then, for each $w \in \mathrm{dom}\, r$ such that $r\uparrow_2(w) = x$, we have an accepting run of A' on $r\uparrow_1(w)$. By a suitable substitution, we obtain an accepting run of A' on b.

□

Lemma 3.5 *Let $A(x, \vec{z})$ be an automaton with an index n weakened by $\emptyset$. Then there exists an automaton $A''[\vec{z}]$ with index n weakened by $\emptyset$, such that, for any semi–algebra $\mathcal{B}$, $A^{\mathcal{B}}[\vec{z}] = \nu x.A^{\mathcal{B}}[x, \vec{z}]$. Moreover, if additionally, the index of A is weakened "by all", so is the index of A'' and, if A is a chain automaton, so is A''.*

Proof of the lemma. Let $A = \langle Sig, Q, \{x, z_1, \ldots, z_m\}, V_I, q_0, Tr, Acc\rangle$, and let

$$Acc = \{(L_1, U_1), \ldots, (L_{n-1}, U_{n-1}), (\emptyset, U_n)\}$$

We start with the following observation : if $x \in V_I$ then, for each $D_1, \ldots, D_m \subseteq B$, $\nu x.A^{\mathcal{B}}[x, \vec{z} \mapsto \vec{D}] = B$. For, it is enough to verify that B is a *fixed point* in this case, and this is obvious. It is plain how to construct an automaton with the required index which, independently of a valuation, recognizes all the elements of any operationally complete semi–algebra (recall that, by Proviso 1, all our semi–algebras are operationally complete).

It remains to consider the case when x is not an initial variable.

We define an automaton A'' similar to one considered in the previous lemma, $A'' = \langle Sig, Q \cup \{x\}, \{z_1, \ldots, z_m\}, V_I, q_0, Tr'', Acc''\rangle$, where

- $Tr'' = Tr \cup \{x = f(y_1, \ldots, y_{\rho(f)}) : q_0 = f(y_1, \ldots, y_{\rho(f)})$ is a transition in $Tr\}$ (exactly as in the previous case),

- $Acc'' = \{(L_1, U_1 \cup \{x\}), \ldots, (L_{n-1}, U_{n-1} \cup \{x\}), (\emptyset, U_n \cup \{x\})\}$.

Let $\mathcal{B}$ be an arbitrary semi–algebra and let $\vec{z} \mapsto \vec{D}$ be an arbitrary valuation, where $D_1, \ldots, D_m \subseteq B$. It remains to verify that $A''^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$ is the greatest fixed point of the mapping

$$M \mapsto A^{\mathcal{B}}[x \mapsto M, \vec{z} \mapsto \vec{D}]$$

where M ranges over $\wp B$. By the Knaster–Tarski Theorem, it is enough to show two things :

(i) $(\forall M \subseteq B)\ M \subseteq A^{\mathcal{B}}[x \mapsto M, \vec{z} \mapsto \vec{D}]$ implies $M \subseteq A''^{\mathcal{B}}[\vec{z} \mapsto \vec{D}]$,

(ii) $A'^{\mathcal{B}}[\vec{z} \mapsto \vec{D}] \subseteq A^{\mathcal{B}}[x \mapsto A'^{\mathcal{B}}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}]$

Again, the argument is analogous to the one given for a similar claim in *Step 1* of the proof of Theorem 3.2, and will be omitted. □

Lemma 3.6 *For automata $A(x_1, \ldots, x_k, \vec{z})$, $A_1(\vec{z})$, $\ldots$, $A_k(\vec{z})$, there exists an automaton $C(\vec{z})$ such that, for any algebra $\mathcal{B}$,*

$$C^{\mathcal{B}}[\vec{z}] = A^{\mathcal{B}}[x_1 \mapsto A_1^{\mathcal{B}}[\vec{z}], \ldots, x_k \mapsto A_k^{\mathcal{B}}[\vec{z}], \vec{z}]$$

Moreover, if the automata $A, A_1, \ldots, A_k$ have all the indices not greater than n then C can be chosen of index n, and if additionally the same weakening applies to all these automata, it is preserved by the construction. Also, the chain form can be keept in the construction.

Note: the notation $A^{\mathcal{B}}[x_1 \mapsto A_1^{\mathcal{B}}[\vec{z}], \ldots, x_k \mapsto A_k^{\mathcal{B}}[\vec{z}], \vec{z}]$ indicates a mapping that sends a tuple $D_1, \ldots, D_m \subseteq B$ onto $A^{\mathcal{B}}[x_1 \mapsto A_1^{\mathcal{B}}[\vec{z} \mapsto \vec{D}], \ldots, x_k \mapsto A_k^{\mathcal{B}}[\vec{z} \mapsto \vec{D}], \vec{z} \mapsto \vec{D}]$.

Proof of the lemma. The construction is straightforward and will be omitted. □

Proof of the Theorem 3.3. We proceed by induction on n.

Recall that Σ_0^μ is the least set of terms containing the basic terms x_i, $x_1 \vee x_2$ and $f(x_1, \ldots, x_{\rho(f)})$, for $f \in Sig$, and closed under substitution. It is plain how to construct automata equivalent to basic terms. For example, an automaton equivalent to the term $f(x_1, \ldots, x_{\rho(f)})$ will have one state, say x_0, and one transition $x_0 = f(x_1, \ldots, x_{\rho(f)})$ $(x_1, \ldots, x_{\rho(f)})$ are variables). No infinite paths are needed in accepting runs of these automata and so we can assume that the acceptance conditions are empty. Thus, by Lemma 3.6, we obtain the claim for all terms τ in $\Sigma_0^\mu = \Pi_0^\mu$.

In order to settle the case $n = 0$, it remains to show that, for any term $\tau(\vec{z}) \in \Sigma_1^\mu$, we can find an equivalent automaton with *empty* acceptance condition, and, for any term $\tau(\vec{z}) \in \Pi_1^\mu$, we can find such an automaton with trivial acceptance condition $(\emptyset, Q)$. The first claim follows directly from Lemmas 3.4 and 3.6. For the second claim first note that the empty acceptance condition is equivalent to $\{(\emptyset, \emptyset)\}$. Then the claim follows easily from Lemma 3.5 and its proof, together with Lemma 3.6. This settles the case of $n = 0$.

Now suppose $n > 0$ and the claim holds for all $n' < n$.

Consider first the class Σ_{2n}^μ. By induction hypothesis, for each term in Π_{2n-1}^μ, we have already an equivalent automaton in chain form with index n weakened both by $\emptyset$ and "by all". Thus the claim for all $\tau \in \Sigma_{2n}^\mu$ follows from Lemmas 3.4 and 3.6.

Now consider the class Π^μ_{2n}. By induction hypothesis, for each term in Σ^μ_{2n-1}, we have an equivalent automaton in chain form with index $n-1$. For each such automaton, we can extend the acceptance condition by a void pair $(\emptyset, \emptyset)$ obtaining thereby an equivalent automaton with index n weakened by $\emptyset$. Thus the claim for the class Π^μ_{2n} follows from Lemmas 3.5 and 3.6.

The claims for the classes Σ^μ_{2n+1} and Π^μ_{2n+1} follow from the just proven facts in the similar way. $\square$

Note on complexity

The above proof gives us an upper bound on the size of the constructed automaton A. Indeed, in Lemmas 3.4 and 3.5 the number of states increases by 1, and in Lemma 3.6, the number of states of the automaton B is not greater than the sum of the corresponding numbers for A, A_1, ..., A_k reduced by k. Then, it is easy to see that an automaton A equivalent to a term τ can be chosen with no more than $|\tau|$ states, where $|\tau|$ denotes the length of term τ.

Concluding remark

We can summarize the above considerations as follows. The class of operations definable by fixed point terms over a given signature interpreted in powerset algebras coincides with the class of operations definable by Rabin automata with variables. Transformations in both directions can be done uniformly for all powerset algebras and are computable, although they apparently differ in complexity : a translation from fixed point terms to automata is linear while the best bound we are able to give for the size of a (vectorial) expression equivalent to a given automaton is exponential.

Moreover, the transformations establish a tight correspondence between the levels of the index hierarchy of automata and those of the fixed point hierarchy, as indicated precisely in Theorems 3.2 and 3.3.

Fixed point terms vs automata with the Mostowski acceptance condition

As we have remarked in Theorem 3.3, an automaton resulting from a fixed point term has its acceptance condition in chain form, which is readily equivalent to the Mostowski acceptance condition.

Construction is even more direct if, instead of a fixed point term, we start with a vectorial fixed point expression (c.f. Proposition 2.11). Indeed, consider a vectorial expression of the form

$$\eta \vec{y_n}. \ldots .\mu \vec{y_1}.\nu \vec{y_0}. (\sigma_1, \ldots, \sigma_k)$$

where η is μ or ν depending on whether n is odd or even, and each σ_i is in some normal form; say, it is a disjunction $\sigma_i = \delta_{i,1} \vee \ldots \vee \delta_{i,\ell_i}$ of atomic terms of the form $f(x_1, \ldots, x_{\rho(f)})$ (here the x_j's are among the variables $\vec{y_0}, \vec{y_1}, \ldots, \vec{y_n}$ plus the free variables of the expression, if any). Then it is not difficult to see that our expression is semantically equivalent (*modulo* initial state) to a Mostowski automaton the states of which are just all the bound variables of the expression, the set of transitions consists of all the equations $y_{p,i} = \delta_{i,j}$ (with $p = 0, 1, \ldots, n$, $i = 1, \ldots, k$ and $j = 1, \ldots, \ell_i$), and the ranking function is defined by *rank* : $y_{p,i} \mapsto p$. Note that in this case, the expression and an equivalent automaton can be viewed as essentially the same object, and the message of Theorem 3.3 is that the two semantics of this object : fixed point semantics and automata semantics, are equivalent.

Conversely, one can carry out a direct translation of Mostowski automata to vectorial fixed point expressions. In such a translation, the states with odd rank would be turned into variables bound by μ while the states with even rank would be turned into variables bound by ν. In order to get a vectorial expression one may need to add some dummy variables, but the size of the resulting expression is polynomial (in fact, $O(n^2)$) in the size of the automaton. Therefore, the vectorial fixed point expressions and Mostowski automata can be viewed as objects of the same level of succinctness, in contrast to vectorial expressions vs Rabin automata (c.f. *Note on complexity* above). Note however that the above does not apply to Mostowski automata vs fixed point terms, due to the cost of converting a vectorial expression into a term.

3.3 Iterative characterization of the fixed point definable tree languages

In this section we discuss an alternative characterization of the fixed point hierarchy in the *powerset tree algebra*. More specifically, we focus on the *sets* of trees definable by closed fixed point terms interpreted in $\wp\mathcal{T}$. We show that tree languages of the subsequent levels of the fixed point hierarchy can be constructed from languages of the preceding levels by means of two operators that can be viewed as somehow analogous to the language–theoretic operators L^* and L^ω.

Compositionality is one of the advantages of the fixed point notation : the expressions denoting complex objects can be naturally decomposed into simpler expressions, denoting simpler objects. However, contrary to the iteration operators $*$ and ω of classical regular (or ω–regular) expressions, the μ and ν operators do not behave as algebraic operations, for obvious reasons. Therefore, a fixed point definition of a tree language does not provide *per se* a decomposition of this language into simpler languages.

This can be remedied as follows. The first observation is that an m–ary operator defined by a fixed point term $\tau(z_1, \ldots, z_m)$ in an algebra $\wp\mathcal{T}_{Sig}$ can be completely characterized by a tree language defined by the same term in the powerset tree algebra over a signature $Sig \cup \{z_1, \ldots, z_m\}$, where the z_i's are considered as constant symbols. The following fact, analogous to Proposition 1.5, can be either deduced from that proposition and Theorems 3.2 and 3.3, or proved directly by induction on the structure of fixed point term. (Recall that for $L \subseteq \mathcal{T}_{Sig \cup \{z_1,\ldots,z_m\}}$, $L[\vec{z} \leftarrow \ldots]$ is the substitution operation described in Section 1.1.)

Proposition 3.7 *Let* $\tau(z_1, \ldots, z_m)$ *be a fixed point term over a signature* Sig*. Then, for any* $K_1, \ldots, K_m \subseteq T_{Sig}$*,*

$$\tau^{\wp\mathcal{T}_{Sig}}[\vec{z} \mapsto \vec{K}] \quad = \quad \tau^{\wp\mathcal{T}_{Sig \cup \{z_1,\ldots,z_m\}}}[\vec{z} \leftarrow \vec{K}]$$

□

Note. In [36], we have stated this fact in a more general form, by extending the concept of substitution operation $L[\vec{z} \leftarrow \ldots]$ to that of an *interpretation* of a tree language L in an arbitrary μ–algebra $\mathcal{A}$. Then, an interpretation of a fixed point term τ in algebra $\mathcal{A}$ can be characterized by the interpretation of the tree language $\tau^{\wp \mathcal{T}_{Sig \cup \{z_1,\ldots,z_m\}}}$ in $\mathcal{A}$.

In order to show that a tree language defined by a fixed point term can be decomposed into languages defined by its subterms, we need to be able to interpret the fixed point operators as operations on languages. Let $L \subseteq \mathcal{T}_{Sig \cup \{x\}}$, where x is a constant symbol not in Sig. Consider an equation

$$X \quad = \quad L[x \leftarrow X]$$

where X ranges over $\wp \mathcal{T}_{Sig}$. We shall show that the extremal fixed points of this equation can be characterized by two operations : an unrestricted and a well–founded iteration.

An intuitive idea behind these operations is simple. Pick up a tree t in L. Next, for each leaf of t labeled by x, pick up again some tree in L and substitute it at this leaf. The resulting tree can still have some leaves labeled by x, so continue the substitutions again and again. This process may be of course infinite, but, by the limit construction, it eventually produces a tree in $\mathcal{T}_{Sig}$. We have informally described unrestricted iteration. For the process of well–founded iteration, we additionally require that only a finite number of substitutions can be made along each path (although the total number of substitutions can be, of course, infinite).

Formally, it is convenient to describe the iteration processes by trees. Let L be as above and let $t \in \mathcal{T}_{Sig}$. Let $D : \operatorname{dom} D \to L$ be a tree valued in L, with $\operatorname{dom} D \subseteq (\operatorname{dom} t)^*$. We recall the operation *last* that has been used in the proof of Theorem 3.2. For $\alpha \in \operatorname{dom} D$,

$$\begin{array}{rcll} \operatorname{last}(\alpha) & = & w & \text{if } \alpha = vw \text{ with } w \in \operatorname{dom} t,\ v \in (\operatorname{dom} t)^* \\ & = & \epsilon & \text{if } \alpha = \epsilon \end{array}$$

A tree D as above is called a *decomposition* of t by L, if, for each $\alpha \in \operatorname{dom} D$ with $\operatorname{last}(\alpha) = w$ and $D(\alpha) = s$, the subtree $t.w$ of t is in $s[x \leftarrow \mathcal{T}_{Sig}]$ and the set of immediate successors of α in $\operatorname{dom} D$ is precisely $\{\alpha\langle wv\rangle \ : \ s(v) = x\}$. A decomposition D is *well–founded* if it is well–founded as a tree, *i.e.* does not contain an infinite path. (Note that a well–founded decomposition may be infinite as its branching degree is not necessarily finite.)

Let

$$\begin{array}{rcll} \operatorname{Iter}_{\infty}(L,x) & = & \{t \in \mathcal{T}_{Sig} \ : & \text{there exists a decomposition of } t \text{ by } L\ \} \\ \operatorname{Iter}_{<\omega}(L,x) & = & \{t \in \mathcal{T}_{Sig} \ : & \text{there exists a well–founded decomposition of } t \text{ by } L\ \} \end{array}$$

Proposition 3.8 *The least and the greatest solutions in $\wp \mathcal{T}_{Sig}$ of the equation $X = L[x \leftarrow X]$, are respectively $Iter_{<\omega}(L,x)$ and $Iter_{\infty}(L,x)$.*

Proof. The arguments are quite analogous as in the similar cases considered in this section. We shall give an argument only for the least fixed point.

Let us abbreviate $\operatorname{Iter}_{<\omega}(L,x) = I$ and $\mathcal{T}_{Sig} = \mathcal{T}$.

By the Knaster–Tarski Theorem, it is enough to show two things :

(i) for all $M \subseteq \mathcal{T}$, $L[x \leftarrow M] \subseteq M$ implies $I \subseteq M$,

(ii) $L[x \leftarrow I] \subseteq I$.

Ad (i). Let $t \in I$, and let D be a well–founded decomposition of t by L. Using Tree Induction Principle, we show that, for each $\alpha \in \mathrm{dom} D$, the subtree $t.\mathrm{last}(\alpha)$ of t is in M. Let $D(\alpha) = s \in L$ and suppose that the claim holds for all immediate successors of α. Then $t.w \in s[x \leftarrow M]$, hence, by assumption, $t.w \in M$.

Ad (ii). Let $t \in s[x \leftarrow I]$ with $s \in L$. Then, for each $w \in \mathrm{dom}\, s$ such that $s(w) = x$, we have a well–founded decomposition of $t.w$ by L. From this, we can easily compose a decomposition D of t by L, with $D(\epsilon) = s$. □

In order to characterize the levels of the fixed point hierarchy over signature Sig, we have to consider tree languages over the extensions of Sig by arbitrary large finite sets of constants. Let us fix a list of symbols $x_1, x_2, \ldots$. Let, for $n < \omega$, $\Sigma^\mu_n(\wp\mathcal{T}_{Sig}*)_0$ be the class of all tree languages L that are in $\Sigma^\mu_n(\wp\mathcal{T}_{Sig \cup \{x_1,\ldots,x_m\}})$, for some m; let $\Pi^\mu_n(\wp\mathcal{T}_{Sig}*)_0$ be defined similarly. (The last index 0 indicates that we consider only 0–ary operations here.)

We are ready to state the following.

Proposition 3.9 *1. $\Sigma^\mu_0(\wp\mathcal{T}_{Sig}*)_0 = \Pi^\mu_0(\wp\mathcal{T}_{Sig}*)_0$ is the family of all finite sets of finite trees over signatures* $\mathrm{Sig} \cup \{x_1, \ldots, x_m\}$, *$m < \omega$;*

2. $\Sigma^\mu_{n+1}(\wp\mathcal{T}_{Sig}*)_0$ is the least class of tree languages containing $\Pi^\mu_n(\wp\mathcal{T}_{Sig}*)_0$ and closed under substitution and under well–founded iteration; that is,

- *if $z_1, \ldots, z_m$ are among $x_1, x_2, \ldots$ and $L, K_1, \ldots, K_m$ are in $\Sigma^\mu_{n+1}(\wp\mathcal{T}_{Sig}*)_0$, so is $L[\vec{z} \leftarrow \vec{K}]$,*
- *if L is $\Sigma^\mu_{n+1}(\wp\mathcal{T}_{Sig}*)_0$, so is $Iter_{<\omega}(L, x_i)$, for $i < \omega$;*

3. $\Pi^\mu_{n+1}(\wp\mathcal{T}_{Sig}*)_0$ is the least class of tree languages containing $\Sigma^\mu_n(\wp\mathcal{T}_{Sig}*)_0$ and closed under substitution and under non–restricted iteration.

Proof. The claim 1 is obvious. The remaining claims follow from Propositions 3.7 and 3.8. □

Corollary 3.10 *Any tree language $L \subseteq \mathcal{T}_{Sig}$ definable by a a fixed point term (or, equivalently, by a Rabin automaton), can be obtained from a finite number of finite trees over* Sig, *by means of the operations of substitution, unrestricted iteration, and the well–founded iteration.*

Note. A characterization by a kind of regular expressions of tree languages recognizable by Büchi automata has been previously shown by Takahashi[47]; it can be viewed as the case of $\Pi^\mu_2(\wp\mathcal{T}_{Sig}*)_0$ in Proposition3.9 above. Another concept of "regular expressions for tree languages" has been proposed by Mostowski [28]. Rather than two iteration operators, Mostowski considers one such operator equipped with a restriction on infinite paths similar to the Mostowski acceptance condition and shows a characterization analogous to Corollary 3.10.

Connection with infinite games

The operations $\mathrm{Iter}_{<\omega}(L, x)$ and $\mathrm{Iter}_{\infty}(L, x)$ discussed above can be naturally characterized in terms of infinite games. Let L be an arbitrary subset of $T_{Sig \cup \{x\}}$, and let $t \in T_{Sig}$. We may consider a game of two players of which one wants to show that t can be decomposed by L and the other strives the opposite. At each moment of the game, if player I has to play, some target node of t is fixed; initially this is the root ϵ. Suppose that $w \in \mathrm{dom}\, t$ is a target node and player I has to play. Then she tries to "cover" (at least partially) the subtree $t.w$ by a tree in L; that is, she picks up a tree s in L such that $t.w \in s[x \leftarrow \mathcal{T}_{Sig}]$ (if there is no such tree, player I looses). Then player I sends the set of nodes $\{wv : v \in \mathrm{dom}\, s \text{ and } s(v) = x\}$ to player II. He answers by choosing one node of this set; this will be the target node for the subsequent play. Note that the set of nodes received by player II may be empty, in this case he looses. It remains to say who wins the game if it lasts to infinity; actually we can define two games according to this decision : in the game, say, $G_{\infty}(L, t)$, we settle that at infinity player I wins, and in the game $G_{<\omega}(L, t)$, player II is the winner. Then it follows directly from the definitions that player I has a winning strategy in the game $G_{\infty}(L, t)$ (respectively, $G_{<\omega}(L, t)$) iff there exists a (well–founded) decomposition of t by L.

A simple but interesting observation is that both games are determined, that is, if there is no suitable decomposition of t by L then player II can always win (a construction of a winning strategy is straightforward). It is interesting, because this fact does not appear to be a corollary to the Martin Determinacy Theorem about Borel games, since the set L may be arbitrary. It can be also observed that if there is a winning strategy for either player, it can be made "memory–less", that is, such that at each moment the next move depends only on the actual position and not on the history of the play until now. Using these observations and the characterization of fixed point terms by the iteration operators, one could inductively define more refined games suitable for characterization of arbitrary fixed point terms; in these games more than one label (like x) would be involved and the winning condition would depend on the labels occurring infinitely often, yet the "memory–less" property would be preserved. Constructions of that kind have been actually carried on by the authors that introduced (independently) automata with Mostowski condition [29, 30, 13] (c.f. *Historical note on automata* above), but without explicit reference to the "basic" games that we have described above.

4 Initiality of the powerset tree algebra

In this section, we prove that the powerset algebra of syntactic trees over a signature *Sig*, $\wp\mathcal{T}_{Sig}$, is initial in the class of all powerset algebras over signature *Sig*, in the following sense : for any semi–algebra $\mathcal{B}$, there is a mapping from $\wp T_{Sig}$ to $\wp B$, which is a morphism w.r.t. fixed point terms. (In the language of category theory, such a property should be rather called *weak* initiality because we do not show that the morphism in consideration is unique.) We apply this result in order to show that the emptiness problem for fixed point terms over the powerset algebra of trees can be reduced to the analogous problem over the powerset algebra of a trivial (one element) algebra, from which we deduce a polynomial time algorithm for this problem.

We need a definition first.

Let $\mathcal{B} = \langle B, \{f^{\mathcal{B}} : f \in Sig\}\rangle$ be a semi–algebra, let $t \in T_{Sig}$, and $b \in B$. An *expansion of b by t* is a tree $D : \mathrm{dom}\, D \to B$, such that $\mathrm{dom}\, D = \mathrm{dom}\, t$, $D(\epsilon) = b$, and, for each $w \in \mathrm{dom}\, D$, if $t(w) = f$, with $\rho(f) = k$, then $D(w) = f^{\mathcal{B}}(D(w1), \ldots, D(wk))$.

Let

$$\mathrm{Exp}\ (t, \mathcal{B}) \quad = \quad \{b \in B \ : \ \text{there exists an expansion of } b \text{ by } t\ \}$$

For $L \subseteq T_{Sig}$, let

$$\mathrm{Exp}\ (L, \mathcal{B}) \quad = \quad \bigcup_{t \in L} \mathrm{Exp}\ (t, \mathcal{B})$$

Note that the operational completeness of a semi–algebra $\mathcal{B}$ (Proviso 1, page 7) implies that, for any $b \in B$, there is always an expansion along some tree in T_{Sig}, in particular $\mathrm{Exp}\ (T_{Sig}, \mathcal{B}) \neq \emptyset$.

We are ready to state the main result of this section.

Theorem 4.1 *Let $\mathcal{B}$ be a semi–algebra and let a mapping $h : \wp(T_{Sig}) \to \wp(B)$ be defined by $h(L) = Exp\ (L, \mathcal{B})$. Then, for any fixed point term over* Sig*, $\tau(z_1, \ldots, z_m)$, and any $L_1, \ldots, L_m \subseteq T_{Sig}$,*

$$h(\tau^{\wp \mathcal{T}_{Sig}}[z_1 \mapsto L_1, \ldots, z_m \mapsto L_m]) \quad = \quad \tau^{\wp \mathcal{B}}[z_1 \mapsto h(L_1), \ldots, z_m \mapsto h(L_m)]$$

Note. It is *not* always true that h is *onto* $\wp B$, as it can be possible that two distinct elements of B have expansions along exactly the same trees and therefore cannot be separated in any value of h. However, any subset of B definable by some term, i.e. $B \supseteq C = \tau^{\wp \mathcal{B}}$, is a value of h.

Proof of the theorem. We shall apply a characterization of fixed point terms by automata of Theorem 3.3. By that result, there exists a Rabin automaton with variables $A(z_1, \ldots, z_m)$, such that, for any semi–algebra $\mathcal{B}$, $\tau^{\wp \mathcal{B}}[z \mapsto h(\vec{L})] = A^{\mathcal{B}}[z \mapsto h(\vec{L})]$ (here $h(\vec{L})$ abbreviates the vector $(h(L_1), \ldots, h(L_k))$).

Thus, it is enough to prove that, for any $L_1, \ldots, L_m \subseteq T_{Sig}$, it holds

$$h(A^{\mathcal{T}}[\vec{z} \mapsto \vec{L}]) \quad = \quad A^{\mathcal{B}}[\vec{z} \mapsto h(\vec{L})]$$

Let $A = \langle Sig, Q, V, V_I, q_0, Tr, Acc\rangle$ (note that A may have initial variables, c.f. page 37). Let $V = \{z_1, \ldots, z_m\}$.

Ad "$\subseteq$". Let $b \in B$, and let t be a tree in $A^{\mathcal{T}}[\vec{z} \mapsto \vec{L}]$, such that there exists an expansion of b by t, say D. Let $r : \mathrm{dom}\, r \to T_{Sig} \times (Tr \cup V)$ be an accepting run of A on t, w.r.t. a valuation $\vec{z} \mapsto \vec{L}$. Recall (Section 1.3, page 15) that $\mathrm{dom}\, r = \mathrm{dom}\, t$ and, for each $w \in \mathrm{dom}\, r$, $r \uparrow_1 (w) = t.w$. We need to construct an accepting run of A on b, w.r.t. a valuation $\vec{z} \mapsto h(\vec{L})$. We define a tree $r' : \mathrm{dom}\, r' \to B \times (Tr \cup V)$, by $\mathrm{dom}\, r' = \mathrm{dom}\, r$ and, for $w \in \mathrm{dom}\, r'$,

$$r'(w) \quad = \quad \langle D(w), \pi_2 \circ r(w)\rangle$$

Note that for each $w \in \operatorname{dom} r'$, if the second component $\pi_2 \circ r'(w)$ is a transition $y = f(x_1, \ldots, x_k)$ then $t(w) = f$ and hence $\pi_1 \circ r'(w) \doteq f^{\mathcal{B}}(\pi_1 \circ r'(w1), \ldots, \pi_1 \circ r'(wk))$, since D is an expansion by t, and if $\pi_2 \circ r'(w) = z_i$ then $t.w \in L_i$ and, as $D(w)$ has obviously an expansion by $t.w$, we have $D(w) \in h(L_j)$. Therefore, r' is a run of A on b w.r.t. the valuation $\vec{z} \mapsto h(\vec{L})$. Moreover, as the state parts of the runs r and r' coincide, this run is accepting.

Ad "$\supseteq$". Now let $b \in B$ and let r be an accepting run of A on b w.r.t. the valuation $\vec{z} \mapsto h(\vec{L})$. Consider for a moment a tree t' over a signature $Sig \cup \{z_1, \ldots, z_m\}$, where $\rho(z_i) = 0$, defined by $\operatorname{dom} t' = \operatorname{dom} r$, and, for each $w \in \operatorname{dom} t'$, if $\pi_2 \circ r(w)$ is a transition $y = f(\vec{x})$ then $t'(w) = f$, and if $\pi_2 \circ r(w) = z_i$ then $t'(w) = z_i$, too. Note that in this last case, there is an expansion of the element $r \uparrow_1 (w)$ by some tree in L_i; let us fix such a tree t_w and a corresponding expansion D_w; let $pick : w \mapsto t_w$ and $pick_D : w \mapsto D_w$ denote the induced mappings. Let $D' = r \uparrow_1$ be the element part of the run r. We define a tree D by substitution $D = D'[pick_D]$, and a tree t by substitution $t = t'[pick]$. Then it is easy to see that D is an expansion of our b by the tree t. Now let $r'' : \operatorname{dom} r \to T_{Sig} \times (Tr \cup V)$ be defined by $\operatorname{dom} r'' = \operatorname{dom} r$, and, for $w \in \operatorname{dom} r''$, $r''(w) = \langle t.w, \pi_2 \circ r(w) \rangle$. Then it is straightforward to verify that r'' is an accepting run of A on t, w.r.t. the valuation $\vec{z} \mapsto \vec{L}$. Therefore $t \in A^{\wp \mathcal{T}_{Sig}}[\vec{z} \mapsto \vec{L}]$, and hence $b \in h(A^{\wp \mathcal{T}_{Sig}}[\vec{z} \mapsto \vec{L}])$.

This remark completes the proof. $\square$

Remark. One may think that the easiest way to show that a mapping between two powerset algebras preserves fixed point terms, should be rather to show that it behaves well w.r.t. the underlying subset ordering. Considering a characterization of extremal fixed points in Theorem 2.2, one would like to require that a candidate for a morphism should preserve the least upper bounds of increasing chains and the greatest lower bounds of decreasing chains. The last, however, is not true about the mapping h considered above.

Let, for example, a signature Sig consist of two symbols : a unary symbol f and a constant symbol c. Let $L_n = \{f^m(c) : m \geq n\}$. Then $\bigcap_{n<\omega} L_n = \emptyset$. Now let $\mathcal{B}$ be an algebra with the universe B consisting of one element, say $B = \{1\}$, and let $f^{\mathcal{B}}(1) = 1$, $c^{\mathcal{B}} = 1$. Then clearly 1 has an expansion in $\mathcal{B}$ by any $f^m(c)$, hence $\operatorname{Exp}(L_n, \mathcal{B}) = \{1\}$, for each n, and consequently $\bigcap_{n<\omega} = \operatorname{Exp}(L_n, \mathcal{B}) = \{1\}$.

In view of this remark, we believe that the use of Theorem 3.3 (or some similar automata–like characterization) in the proof of Theorem 4.1 is essential.

4.1 Internalization

We apply Theorem 4.1 to show a close connection between the algebra $\wp \mathcal{T}_{Sig}$ and an algebra $\mathbf{t}$, for $t \in T_{Sig}$ (c.f. Section 1.2).

Proposition 4.2 *For a closed fixed point term τ and $t \in T_{Sig}$, $t \in \tau^{\wp \mathcal{T}_{Sig}}$ if and only if $\epsilon \in \tau^{\mathbf{t}}$.*

Proof. Let $h : \wp(T_{Sig}) \to \wp(\operatorname{dom} t)$ be the mapping defined in Theorem 4.1. Observe that ϵ (the root of t) considered as an element of the algebra $\mathbf{t}$ has the unique expansion by t. Therefore, for any $L \subseteq T_{Sig}$, $t \in L$ iff $\epsilon \in h(L)$. Thus the claim follows from Theorem 4.1. □

Note. We have first observed this property in [3] (and proved it directly for a stronger version of the fixed point calculus) and called it *internalization*, because it enables us to pass from *external* semantics for the fixed point terms in $\wp\mathcal{T}_{Sig}$ to an *internal* semantics in $\mathbf{t}$ (and *vice versa*). We believe that this phenomenon is very general, since many objects considered in computer science, as words, graphs, trees, automata..., can be either organized into algebras (or logical structures) or treated themselves as algebraic (or logical) structures. The virtue of the fixed point calculus is that the both approaches can be considered in a unified way, just as two legitimate interpretations of the same fixed point term notation.

4.2 Nonemptiness problem

Consider the following decision problems :

> *Satisfiability.* Given a fixed point term without free variables, τ, is there a powerset algebra $\wp\mathcal{B}$ such that $\tau^{\wp\mathcal{B}} \neq \emptyset$?
>
> *Nonemptiness.* For a fixed powerset algebra $\wp\mathcal{B}$, given τ, is $\tau^{\wp\mathcal{B}} \neq \emptyset$?

We are primarily interested in solving this second problem for the powerset algebra of syntactic trees. It turns out however that both problems can be reduced to some simple finite powerset algebra, in fact the simplest one.

In what follows, *Sig* is an arbitrary signature. Let $\mathcal{B}_0$ be an algebra over *Sig* with the universe consisting of only one element, say 1, and all the operations defined by : $f^{\mathcal{B}_0}(1,\ldots,1) = 1$, for $f \in Sig$. The following fact induces a reduction of the satisfiability problem, as well as the nonemptiness problem for $\wp\mathcal{T}_{Sig}$, to the nonemptiness problem for $\wp\mathcal{B}_0$.

Proposition 4.3 *For a fixed point term without free variables, τ, the following conditions are equivalent :*

1. $\tau^{\wp\mathcal{T}_{Sig}} \neq \emptyset$,
2. *there is a semi-algebra $\mathcal{B}$, such that $\tau^{\wp\mathcal{B}} \neq \emptyset$,*
3. $\tau^{\wp\mathcal{B}_0} \neq \emptyset$.

Proof. Implications $2 \Rightarrow 1$ and $3 \Rightarrow 1$ follow directly from Theorem 4.1, implication $1 \Rightarrow 2$ is trivial, and the implication $1 \Rightarrow 3$ follows from Theorem 4.1 and the fact that 1 has an expansion in $\mathcal{B}_0$ by *any* tree $t \in T_{Sig}$. □

We note that the equivalence of the fixed point terms and automata (Theorems 3.2 and 3.3), imply an analogous result for automata :

Proposition 4.4 *For any Rabin automaton without free variables, A, the following conditions are equivalent :*

1. $A^{\wp\mathcal{T}_{Sig}} \neq \emptyset$,

2. *there is a semi–algebra* $\mathcal{B}$, *such that* $A^{\wp\mathcal{B}} \neq \emptyset$,

3. $A^{\wp\mathcal{B}_0} \neq \emptyset$

$\square$

How difficult computationally is the nonemptiness problem for $\wp\mathcal{B}_0$?

The ordered universe of the algebra $\wp\mathcal{B}_0$ can be identified with that of the *Boolean* lattice $\langle\{0,1\},\wedge,\vee\rangle$, and one can translate the fixed point terms over *Sig* to the fixed point terms over the signature $\{\wedge,\vee\}$ in such a way that the values of a closed term and its translation are identical. Namely, $trans(\tau_1 \vee \tau_2) = trans(\tau_1) \vee trans(\tau_2)$, $trans(\eta x.\tau) = \eta x.trans(\tau)$, for $\eta \in \{\mu,\nu\}$ and $trans(f(\tau_1,\ldots,\tau_k))$ $trans(\tau_1) \wedge \ldots \wedge trans(\tau_k)$, for $f \in Sig$.

Now, it is immediate to verify that, for any monotonic Boolean function $g : \{0,1\}^{k+1} \to \{0,1\}$, and for any $b_1,\ldots,b_k \in \{0,1\}$, $\mu y.g(y,b_1,\ldots,b_k) = g(0,b_1,\ldots,b_k)$ and $\nu y.\ g(y,b_1,\ldots,b_k) = g(1,b_1,\ldots,b_k)$. Therefore, evaluation of a closed fixed point term over the Boolean lattice can be reduced to evaluation of an ordinary term built out from $\vee$, $\wedge$, 0 and 1. The last can clearly be done in linear time. As all the reductions are also linear, we can note the following corollary to Proposition 4.3.

Corollary 4.5 *The satisfiability problem for fixed point terms, as well as the nonemptiness problem for fixed point terms interpreted in the powerset tree algebra, can be solved in linear time.*

We cannot consider the above result as particularly optimistic, because our fixed point calculus is not very succint. Note that an upper bound for the emptiness problem for fixed point terms does not imply similar results neither for vectorial fixed point terms, not for the Rabin or Mostowski automata, as the only translations that we know to carry on from fixed point terms to those objects are not polynomial.

Note. A similar result has been obtained recently by Janin and Walukiewicz [21] for a special kind of formulas of the modal μ–calculus (see Section 6) called disjunctive formulas.

5 Hierarchy

In this section we shall examine the problem whether the alternation hierarchy of fixed point terms induces a proper hierarchy of fixed point definable operations in a μ–algebra. We shall see that it is so in the powerset algebra of syntactic trees, under some general hypothesis on the signature. More specifically, we shall exhibit a family of sets of binary trees the definitions of which will require arbitrarily long sequences of alternations of the least and greatest fixed point operators. For simplicity of presentation, we shall first consider languages over different signatures; later on, we shall observe that all these can be encoded over one fixed signature, provided that it contains at least two symbols one of them of an arity at least 2.

Note. The family of examples to be presented has been already shown by this author in [36]; in the actual presentation, we simplify the phrasing according to [13].

In this section we shall deal only with automata without variables running over syntactic trees. Therefore, we shall use the classical definition of an automaton run (page 15). We shall also use a more standard notation $L(A)$ for $A^{\wp\mathcal{T}_{Sig}}$, the set of trees accepted by an automaton A.

Let, for $n = 1, 2, 3, \ldots$, $Sig_n = \{a_1, a_2, \ldots, a_n\}$ be a signature, where the arity of each a_i is 2. We define two sequences of tree languages, $(M_n)_{n\geq 0}$ and $(N_n)_{n\geq 0}$, by the interpretation of two sequences of fixed point terms over signatures Sig_n in the algebras $\wp\mathcal{T}_{Sig_n}$, respectively.

$$\begin{aligned} M_n &= \xi x_n. \ldots \mu x_3.\nu x_2.\mu x_1.a_1(x_1, x_1) \vee a_2(x_2, x_2) \vee \ldots \vee a_n(x_n, x_n) \\ N_n &= \xi' x_n. \ldots \nu x_3.\mu x_2.\nu x_1.a_1(x_1, x_1) \vee a_2(x_2, x_2) \vee \ldots \vee a_n(x_n, x_n) \end{aligned}$$

where $\xi, \xi' \in \{\mu, \nu\}$ and, $\xi = \mu$ iff n is odd iff $\xi' = \nu$.

What do these tree languages look like ?

Let L be any of the above sets M_n or N_n. There is one common property that characterizes the trees t in all these sets L :

> For any infinite path, if i is the greatest index such that a_i occurs infinitely often on this path, then the variable x_i is bound by ν (not by μ) in the above definition of L.

This obviously amounts to the following two properties :

$t \in M_n$ iff, for any infinite path P of t, $\max\{i : a_i \in \mathrm{Inf}(t, P)\}$ is *even*,

and,

$t \in N_n$ iff, for any infinite path P of t, $\max\{i : a_i \in \mathrm{Inf}(t, P)\}$ is *odd*.

The easiest way to see the above characterization is by translation of the fixed point terms into the Rabin automata, e.g. by the method of Theorem 3.3. For $n = 1, 2, \ldots$, consider the following Rabin automata (without variables) : $A_n = \langle Sig_n, \{1, 2, \ldots, n\}, 1, Tr_n, Acc_n\rangle$ and $B_n = \langle Sig_n, \{1, \ldots, n\}, 1, Tr_n, Acc'_n\rangle$, where the numbers $1, 2, \ldots, n$, serve as states of both A_n and B_n, the set of transitions Tr_n, also common to A_n and B_n, consists of all the equations $i = a_j(j, j)$, for $i, j \in \{1, \ldots, n\}$, and the acceptance conditions are specified as follows (here *Even* and *Odd* stand for the sets of even and odd natural numbers respectively, and $[k, m]$ is an abbreviation for $\{j : k \leq j \leq m\}$) :

$$Acc_n = \{(Odd \cap [i+1, n], Even \cap [i, n]) : \ i \text{ is even and } i \leq n\}$$

$$Acc'_n = \{(Even \cap [i+1, n], Odd \cap [i, n]) : \ i \text{ is odd and } i \leq n\}$$

For example,

$$Acc_7 = \{(\{3, 5, 7\}, \{2, 4, 6\}), (\{5, 7\}\{4, 6\}), (\{7\}, \{6\})\}$$

$$Acc'_7 = \{(\{2,4,6\},\{1,3,5,7\}),(\{4,6\},\{3,5,7\}),(\{6\},\{5,7\}),(\emptyset,\{7\})\}$$

Note that $Acc_1 = \emptyset$, and $Acc_1 = \{(\emptyset,\{0\})\}$.

It is easy to see that the above are precisely the automata that we shall obtain from the fixed point terms defining the languages M_n and N_n, according to the method of Theorem 3.3 (up to replacing x_i by just i, for the sake of simplicity). In particular,

$$M_n = L(A_n), \text{ for } n = 0,1,\ldots$$

$$N_n = L(B_n), \text{ for } n = 0,1,\ldots$$

We are ready to state the main result of this section.

Theorem 5.1 *1. For $n < \omega$ odd,*

(a) M_n is in $\Sigma^\mu_n(\mathrm{Sig}_n)$ but not in $\Pi^\mu_n(\mathrm{Sig}_n)$,

(b) N_n is in $\Pi^\mu_n(\mathrm{Sig}_n)$ but not in $\Sigma^\mu_n(\mathrm{Sig}_n)$.

2. For $0 < n < \omega$ even,

(a) M_n is in $\Pi^\mu_n(\mathrm{Sig}_n)$ but not in $\Sigma^\mu_n(\mathrm{Sig}_n)$,

(b) N_n is in $\Sigma^\mu_n(\mathrm{Sig}_n)$ but not in $\Pi^\mu_n(\mathrm{Sig}_n)$.

Proof. The positive part of the claim follows from the definition of the languages M_n and N_n. In order to prove the negative part, we find it more convenient to deal with automata rather than directly with fixed point terms and then we shall use Theorem 3.3. It is enough to prove the following :

Claim 5.2 *1. for $m < \omega$, M_{2m+1} cannot be recognized by a Rabin automaton with index $m+1$ weakened by $\emptyset$ and "by all",*

2. for $m < \omega$, N_{2m+1} cannot be recognized by a Rabin automaton with index m,

3. for $m > 0$, M_{2m} cannot be recognized by a Rabin automaton with index m weakened "by all",

4. for $m > 0$, N_{2m} cannot be recognized by a Rabin automaton with index m weakened by $\emptyset$.

The above suggests that we will have to consider 4 cases. In fact, one can note that the cases 1 and 4 are symmetric as in both sets it is required that a_i with the highest possible i may not occur infinitely often along a path. Similarly, the case 3 is analogous to 2, so we will essentially deal with two cases only.

It will be useful to introduce a concept of a partial tree over a binary signature and a partial run over such a tree. We fix a symbol $\perp$ as a 0–ary (constant) symbol. Let Sig be any of the signatures Sig_n, and let $Sig_\perp$ abbreviate the signature $Sig \cup \{\perp\}$. Clearly $T_{Sig} \subseteq T_{Sig_\perp}$. We shall refer to the trees over $Sig_\perp$ as *partial trees* over Sig; in this context the trees from T_{Sig} will be emphatically called *total.*

Let $A = \langle Sig, Q, q_0, Tr, Acc \rangle$ be a Rabin automaton (over signature Sig) and let $q \in Q$. Let $t \in T_{Sig_\perp}$ be a partial tree. A *partial q–run* of A on t is any tree $r : \mathrm{dom}\, t \longrightarrow Q$ such that $r(\epsilon) = q$, and, for each $w \in \mathrm{dom}\, t$ such that $t(w) \neq \perp$, say $t(w) = a$, $r(w) = a(r(w1), r(w2))$ is a transition in Tr. (Note that we do *not* adapt the automaton A to a new signature $Sig_\perp$; in particular, nothing is imposed on the states occurring at the leaves of a partial run.)

A partial q–run is *accepting* if any of its infinite paths is accepting, in the usual sense (note that $\perp$ may not appear on an infinite path). We say that a tree $t \in T_{Sig_\perp}$ is *partially accepted* by A if there exists a state $q \in Q$ and a a partial accepting q–run of A on t. Reference to q will be sometimes omitted. Note that q need not to be the initial state. We denote by $L_p(A)$ the set of all partial trees partially accepted by A. Clearly, $L(A) \subseteq L_p(A)$; note however that $L_p(A)$ may be nonempty while $L(A) = \emptyset$.

Now, for any total tree $t \in T_{Sig}$, we define the set t^p of partial trees,

$$t^p \;=\; \{t' \in T_{Sig_\perp} \;:\; \text{for some } w \in \mathrm{dom} t,\, t.w \in t'[\perp \leftarrow T_{Sig}]\}$$

Intuitively, this is the set of those all partial trees that occur in t (not necessarily as complete subtrees).

For $M \subseteq T_{Sig}$, we set $M^p = \bigcup_{t \in M} t^p$.

We shall use the following fact, coming easily from the aforementioned characterizations of the sets M_n and N_n.

Observation 5.3 *For $m \leq n$,*

$$\begin{aligned} M_n^p \cap T_{\mathrm{Sig}_m} &= M_m^p \\ N_n^p \cap T_{\mathrm{Sig}_m} &= N_m^p \end{aligned}$$

We call a partial tree $t \in T_{Sig_\perp}$ a *branch* if, for any $w \in \mathrm{dom}\, t$, there is $v \in \mathit{domt}$, $v \geq w$, such that $t(v) = \perp$.

The following is the heart of our proof.

Lemma 5.4 *Let M be any of the sets N_n or M_n, $n = 1, 2, \ldots$. Let A be a Rabin automaton with such an index that hypothesis $L(A) = M$ would contradict the Claim 5.2 above. Suppose $L_p(A) \subseteq M^p$. Then there exists a branch in $M^p - L_p(A)$.*

Proof of the lemma. For $M = M_1$, the statement may appear strange, since $M_1^p = M_1 = \emptyset$, but in this case the claim only says that the inclusion $L_p(A) \subseteq M^p$ is impossible. Indeed, due to our Proviso 2 (see page 14), an automaton with the index 1 weakened both by $\emptyset$ and "by all", that is, with a trivial acceptance condition $(\emptyset, Q)$ must accept at least one tree.

For $M = N_1$, clearly M consists of the unique tree of T_{Sig_1}, $t : \{1, 2\}^* \to \{a_1\}$. On the other hand, an automaton with the empty acceptance condition cannot partially accept any tree with an infinite path. Thus any infinite branch in M^p will do.

Now we are going to show that the claim holds for all the sets N_n. Let $n > 1$ and suppose that the claim holds for all $N_{n'}$, $n' < n$. We shall consider two cases.

Case 1. n is even, say $n = 2m$, for some $m > 0$.

Let $A = \langle Sig_n, Q, q_0, Tr, Acc \rangle$ be an automaton with

$$Acc = \{(L_1, U_1), \ldots, (L_{m-1}, U_{m-1}), (\emptyset, U_m)\}$$

and suppose $L_p(A) \subseteq N_n^p$.

We define an automaton A' over a smaller signature Sig_{n-1} by restricting the transitions to Sig_{n-1} and also simplifying the acceptance condition, namely

$$A' = \langle Sig_{n-1}, Q, q_0, Tr \cap (Q \times Sig_{n-1} \times Q \times Q), Acc' \rangle$$

with $Acc = \{(L_1, U_1), \ldots, (L_{m-1}, U_{m-1})\}$ (if $m = 1$, this acceptance condition is empty). It should be clear that any accepting partial run of A' is also accepting in the sense of A, hence $L_p(A') \subseteq N_n^p$. But, since $L_p(A') \subseteq T_{Sig_{n-1}}$, by Observation 5.3, we have $L_p(A') \subseteq N_{n-1}^p$. Since A' has index $m-1$, by induction hypothesis, there exists a branch $B \in N_{n-1}^p - L_p(A')$. We shall use it to construct a desired branch in $N_n^p - L_p(A)$.

Let k be the cardinality of U_m. We first construct a sequence of partial trees B_0, $B_1, \ldots, B_k$, as follows. Let $\hat{B} = a_n(\perp, \perp)[\perp \leftarrow B]$ (here $a_n(\perp, \perp)$ stands for the tree with the domain $\{\epsilon, 1, 2\}$ and the values at $\epsilon, 1, 2$ being a_n, $\perp$ and $\perp$, respectively; the definition of substitution has been given in Section 1.1). Let

$$B_0 \quad = \quad \hat{B}[\perp \leftarrow a_n(\perp, \perp)]$$

and let, for $i = 0, 1, \ldots, k-1$,

$$B_{i+1} \quad = \quad \hat{B}[\perp \leftarrow B_i]$$

Clearly, each B_i is a branch. Let $B' = B_k$. We claim $B' \in N_n^p - L_p(A)$.

The assertion $B' \in N_n^p$ follows easily from the construction : a_n may occur at most $k+2$ times on a path, and any infinite path of B' is cofinal with some path in B. To prove $B' \notin L_p(A)$ suppose the converse and let $r : \mathrm{dom}\, B' \to Q$ be a partial accepting q–run of A on B'. Consider an "occurrence" of B in B', that is, a node $w \in \mathrm{dom}\, B'$, such that $B'.w \in B[\perp \leftarrow T_{Sig_n}]$. Let $r(w) = p$. Such an occurrence induces a partial p–run of A on B, r' say, defined by $r'(v) = r(wv)$, for $v \in \mathrm{dom}\, B$. But r' can be also considered as a partial run of the automaton A' on B. Now a crucial observation is that r' may not be accepting as a partial run of A', since otherwise we would have $B \in L_p(A')$, contradictory to the choice of B. On the other hand, r' *is* obviously an accepting partial run of A, as a fragment of an accepting partial run. From that we can conclude that some state from U_m must occur as a value of r' at least once (in fact, infinitely often; if $U_m = \emptyset$, we obtain a contradiction which completes the proof in this case). Let $r'(v) \in U_m$. Note that, since B is a branch, there must be a leaf $v_1 \in \mathrm{dom} B$, $v \leq v_1$. As this situation holds for every occurrence of B in B', we easily derive that there exist some nodes $w_1, w_2, w_3 \in \mathrm{dom}\, B'$ such that $w_1 < w_2 < w_3$, $r(w_1) = r(w_3) \in U_m$, and $B'(w_2) = a_n$. This fact will allow us

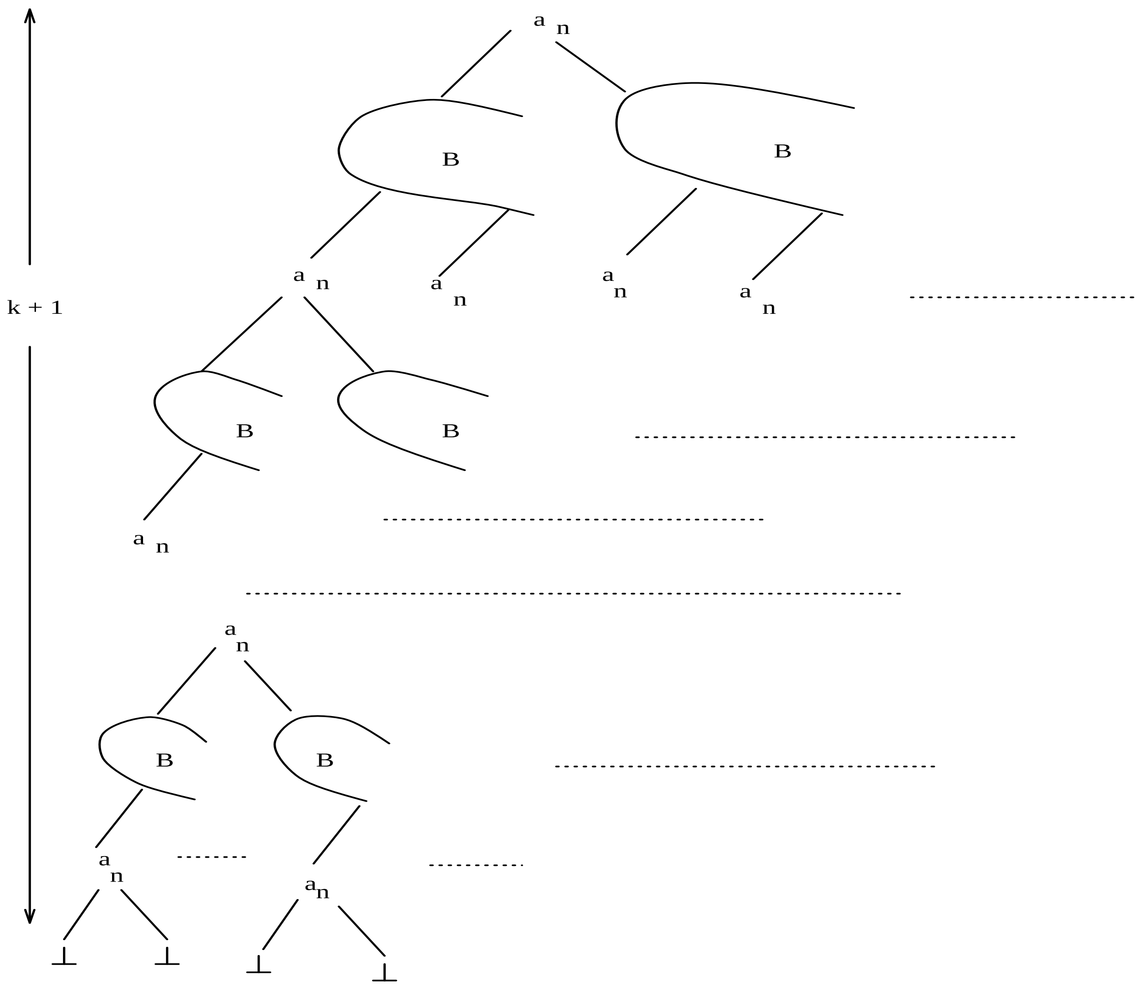

Figure 1: Branch $B' \in N_n^p - L_p(A)$ for even n.

to construct a partial tree that is not in N_n^p but is nevertheless partially accepted by A, contradictory[8] to the assumption $L_p(A) \subseteq N_n^p$.

More specifically, we first define a sequence of partial trees $B'_0, B'_1, \ldots$, a sequence of partial q–runs $r_0, r_1, \ldots$, and, as an auxiliary, a sequence of nodes $v_0, v_1, \ldots$, as follows. We set $v_0 = w_3$, $B'_0 = B'$ and $r_0 = r$.

Let v be such that $w_3 = w_1 v$. For $i < \omega$, we define

$$\begin{aligned} B'_{i+1} &= B'_i[v_i \leftarrow B'.w_1] \\ r_{i+1} &= r_i[v_i \leftarrow r.w_1] \\ v_{i+1} &= v_i v \end{aligned}$$

Let $B'' = \lim B'_n$ and $r'' = \lim r_n$. Clearly, B'' contains a path with infinitely many occurrences of a_n, and then $B'' \notin N_n^p$. On the other hand, it is easy to see that r'' is an accepting partial q–run of A on B''. Indeed, the only infinite path of r'' that is not cofinal with some path of r does contain infinitely many occurrences of a state from U_n and therefore is accepting.

This remark completes the induction step for even n.

Case 2. $n > 1$ is odd, say $n = 2m+1$, for some $m > 0$.

Let $A = \langle Sig_n, Q, q_0, Tr, Acc\rangle$ be an automaton with $Acc = \{(L_1, U_1), \ldots, (L_m, U_m)\}$ and suppose $L_p(A) \subseteq N_n^p$.

Now, for each $i = 1, \ldots, m$, we consider an automaton A'_i over the signature Sig_{n-1}, obtained from A by restriction of Tr to Sig_{n-1}, and moreover dropping out the states from L_i (we may assume without loss of generality that q_0 is not in any L_i). That is,

$$A'_i = \langle Sig_{n-1}, Q_i, q_0, Tr_i, Acc'_i\rangle$$

with $Q_i = Q - L_i$, $Tr_i = Tr \cap (Q_i \times Sig_{n-1} \times Q_i \times Q_i)$, and $Acc_i = \{(L_1 \cap Q_i, U_1 \cap Q_i), \ldots, (L_m \cap Q_i, U_m \cap Q_i)\}$. Note that $L_i \cap Q_i = \emptyset$, and thus each automaton A'_i has index m weakened by $\emptyset$. Again, it is easy to see that $L_p(A'_i) \subseteq N_{n-1}^p$. Next, it is not difficult to construct an automaton, D say, of index m weakened by $\emptyset$, such that $L_p(D) = \bigcup_{i=1,\ldots,m} L_p(A'_i)$. Clearly, $L_p(D) \subseteq N_{n-1}^p$. Therefore, by the induction hypothesis, there exists a branch $B \in N_{n-1}^p - L_p(D)$. Again, we shall use this branch to construct a required branch $B' \in N_n^p - L_p(A)$.

The construction will be similar as in the previous case, but this time B' will be obtained as a limit of an infinite sequence. Hence, for B' to be a branch, we must be more careful when making substitutions. Technically, it will be convenient to introduce for a moment an auxiliary constant symbol $\perp'$ that will be used for substitutions. Let

$$\begin{aligned} \hat{B} &= a_n(\perp, \perp')[\perp' \leftarrow B] \\ B_0 &= \hat{B}[\perp \leftarrow a_n(\perp, \perp')] \end{aligned}$$

[8] Our construction generalizes a pumping argument used by Rabin [42] in order to show, in our terminology, that $N_2 \notin \Pi_2^\mu(Sig_2)$.

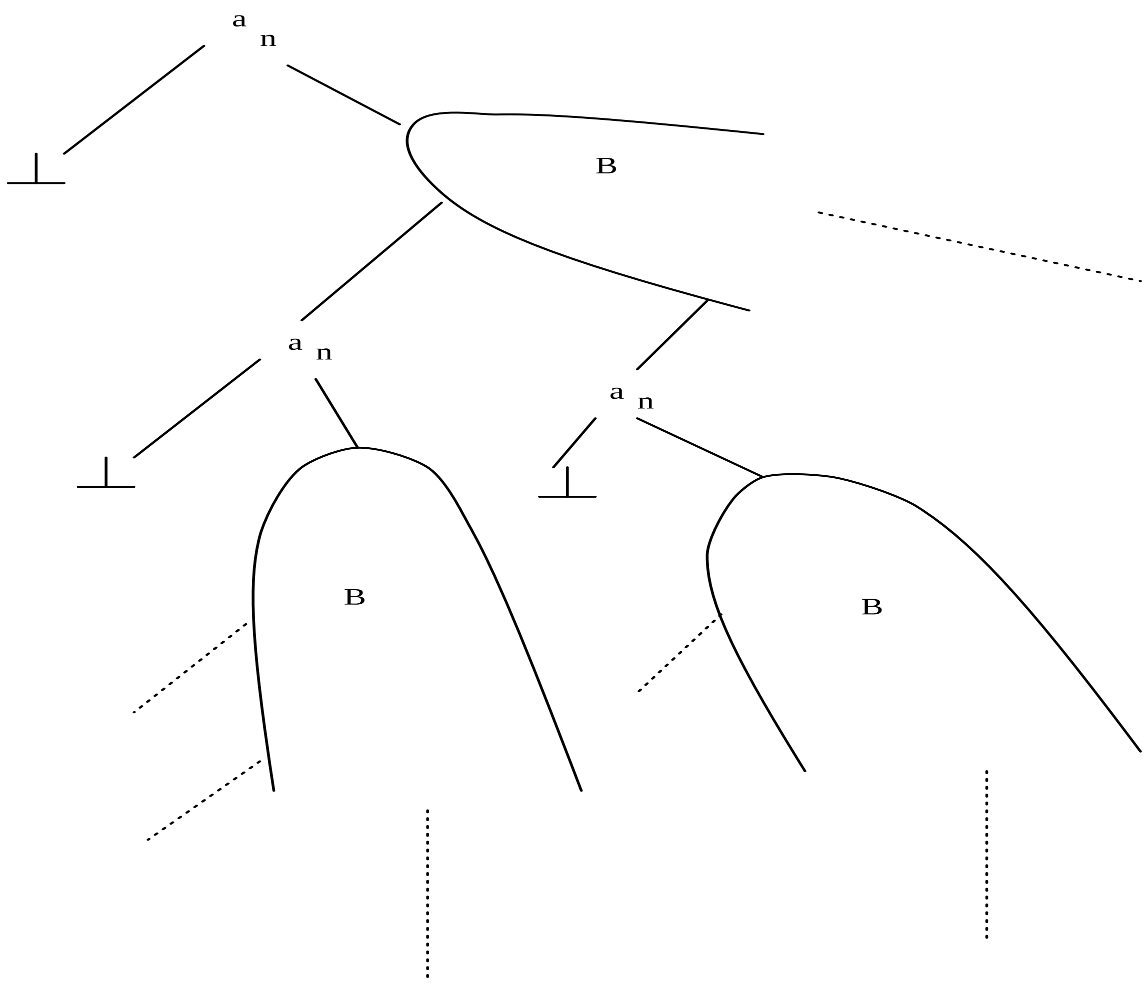

Figure 2: Branch $B' \in N_n^p - L_p(A)$ for odd n.

and let, for $i = 0, 1, 2, \ldots$,

$$B_{i+1} \quad = \quad B_0[\perp' \leftarrow B_i]$$

It is easy to see, that the sequence B_i is convergent and its limit is a partial tree over Sig_n (without $\perp'$). Let $B' = \lim B_i$. We claim $B' \in N_n^p - L_p(A)$.

The assertion $B' \in N_n^p$ follows from the fact that any path of B' that is not cofinal with a path of B, contains infinitely many occurrences of a_n. To prove $B' \notin L_p(A)$ suppose the contrary and let $r : \operatorname{dom} B' \to Q$ be a partial accepting q–run of A on B'. As in the previous case, consider an occurrence of B in B', that is, a node $w \in \operatorname{dom} B'$, such that $B'.w \in B[\perp \leftarrow T_{Sig_n}]$. Let $r(w) = p$. Again, the occurrence of B in B' induces a partial p–run of A on B, r' say, defined by $r'(v) = r(wv)$, for $v \in \operatorname{dom} B$. By choice of B, this r' may not be accepting if considered as a run of an A_i', $i = 1, \ldots, m$. That is, for any i, there must be a path that is accepting for A but not for A_i. But this means that, for any i, we can find a node, say $v \in \operatorname{dom} B$, such that $r'(v) = r(wv) \in L_i$ (if some $L_i = \emptyset$, we obtain a contradiction that ends the proof). Since B is a branch, and by construction of B', the node wv is succeeded by another node in $\operatorname{dom} B'$, which is again an occurrence of B. Since the above observation holds for any occurrence of B in B', and for any $i = 1, \ldots, m$, we can inductively construct an infinite path in $\mathrm{dom}\, B' = \mathrm{dom}\, r$ containing infinitely many occurrences of some state in L_i, for each $i = 1, \ldots, m$, and thus clearly non-accepting. But this contradicts the assumption that r is an accepting run.

This remark completes the proof of the claim of the lemma for all sets N_n, $n = 1, 2, \ldots$.

The proof of the statement for the sets M_n, is almost completely analogous, although the actual argument for M_{2m} will coincide with the argument for N_{2m+1}, and the argument for M_{2m+1} will coincide with the argument for N_{2m}. One more remark must be added for the special case when a trivial pair $(\emptyset, Q)$ appears in the acceptance condition. We sketch the argument briefly.

The case of M_1 is already settled. For $n > 1$, we consider two cases.

Case 1. $n = 2m$, for some $m > 0$.

Let A be an automaton of index m weakened "by all" such that $L_p(A) \subseteq N_n^p$.

We construct the automata A_i', as in Case 2 of the proof for N_n. These new automata have index m weakened by $\emptyset$, but it is plain that their index is also still weakened "by all". Therefore, the induction hypothesis may be applied, and the argument proceeds exactly as in the previous case.

Case 2. $n = 2m + 1$, for some $m > 0$.

Let A be an automaton with the index $m + 1$ weakened both by $\emptyset$ and "by all", such that $L_p(A) \subseteq M_n^p$. Let the acceptance condition of A be

$$Acc = \{(L_1, U_1), \ldots, (L_m, U_m), (\emptyset, U_{m+1})\}$$

and let $L_i \cup U_i = Q$.

If $i = m + 1$, that is $U_{m+1} = Q$, then every infinite path in every run is accepting. In this case, it is fairly easy to construct a branch that may not be accepted by A. For example, let B be defined as follows : $\mathrm{dom}\, B = \{2\}^* \cup \{2\}^*1$, $B(w) = a_1$ for $w \in 2^*$ and $|w| \leq |Q|$, $B(w) = a_2$ for $w \in 2^*$, and $|w| > |Q|$, and $B(w) = \perp$ for $w \in \{2\}^*1$. Clearly, B is a branch in M_n^p. Suppose it is partially accepted by A and let r be a partial accepting q–run. There must be some $0 \leq n_1 < n_2 \leq |Q|$ such that $r(2^{n_1}) = r(2^{n_2})$. Hence, by an obvious pumping construction, we can construct an accepting partial q–run of A on a branch, say B', that differs from B in that $B(w) = a_1$, for all $w \in 2^*$. Clearly, B' is not in M_n^p, which is a contradiction to the assumption $L_p(A) \subseteq M_n^p$.

Now suppose that the i satisfying $L_i \cup U_i = Q$ is different from $m + 1$. We construct an automaton A' as in Case 1 of the proof for N_n. This automaton has now index m weakened "by all" (recall that A had index $m+1$ weakened by $\emptyset$ and "by all"). Therefore, the induction hypothesis may be applied, and the argument proceeds exactly as in the previous case.

This remark completes the proof of the lemma. $\square$

Now we are ready to prove the Claim 5.2.

Ad 1. Suppose that, for some $m < \omega$, the set M_{2m+1} is recognized by a Rabin automaton A with an index $m+1$ weakened by $\emptyset$ and "by all". For $m = 0$, we have $M_1 = \emptyset$, while the automaton A has trivial acceptance condition which, by our Proviso 2 on automata, implies $L(A) \neq \emptyset$; a contradiction.

Suppose $m > 0$. We may assume without loss of generality, that every state of A occurs in some accepting run. Indeed, if this is not the case originally, we can drop out the "useless" states, without increasing the index of the automaton. This assumption implies $L_p(A) = M^p_{2m+1}$. Then, by Lemma 5.4, there exists a branch B in $M^p_{2m+1} - L_p(A)$. Clearly, any branch in M^p_{2m+1} can be extended to a "total" tree in M_{2m+1}. That is, we can find a tree $t \in M_{2m+1}$, and $w \in \operatorname{dom} t$, such that $t.w \in B[\perp \leftarrow T_{Sig_{2m+1}}]$. Let r be an accepting run of A on t and let $r(w) = q$. Then the restriction of r to the occurrence of B in t at the node w is clearly a partial accepting q–run of A on B, thus $B \in L_p(A)$, a contradiction.

The argument for the remaining cases is similar.

The proof of the theorem is now completed. $\square$

We have shown by Theorem 5.1, that, in the algebra $\wp\mathcal{T}_{Sig_n}$, $\Sigma^\mu_n(Sig_n) \neq \Pi^\mu_n(Sig_n)$; consequently $\Sigma^\mu_i(Sig_n) \neq \Sigma^\mu_{i+1}(Sig_n)$ and $\Pi^\mu_i(Sig_n) \neq \Pi^\mu_{i+1}(Sig_n)$, for $0 \leq i < n$, that is, the hierarchy has the height at least n. From this, it is already not difficult to construct a single powerset tree algebra in which the hierarchy is actually infinite.

Theorem 5.5 *Let* $\mathrm{Sig} = \{f, c\}$ *be a signature, where f is binary and c is constant symbol. Then, for all $n < \omega$, the classes* $\Sigma^\mu_n(\mathrm{Sig})$ *and* $\Pi^\mu_n(\mathrm{Sig})$ *are incomparable, that is, the fixed point hierarchy in the algebra* $\wp\mathcal{T}_{Sig}$ *is infinite.*

Proof. The argument will consist in encoding the trees over all signatures Sig_n considered above by the trees over Sig, and then showing that the encoded versions of the sets M_n and N_n require the same indices of Rabin automata as before.

We first fix a sequence of distinct finite trees in T_{Sig}, $b_1, b_2, \ldots$. Let $b_1 = c$ and $b_{i+1} = f(c, c)[c \leftarrow b_i]$, for $i = 1, 2, \ldots$.

Now, for each $t \in \bigcup_n T_{Sig_n}$, we shall define a tree $\hat{t} \in T_{Sig}$. It will be more informative to start with a tree, say t', over an auxiliary signature $\{f, a_1, \ldots, a_n\}$, where $a_1, \ldots, a_n$ are considered as *constant*, not binary, symbols. Let $h : \{1,2\}^* \to \{1,2\}^*$ be a homomorphism induced by the mapping

$$\begin{aligned} 1 &\mapsto 11 \\ 2 &\mapsto 12 \end{aligned}$$

and let $code : \{1,2\}^* \to \{1,2\}^*$ be a mapping given by

$$code(w) = h(w)2$$

We define t' by

$$\operatorname{dom} t' = \{v : v \leq code(w) \text{ for some } w \in \{1,2\}^*\}$$

and

$$\begin{aligned} t'(code(w)) &= t(w) \\ t'(v) &= f \text{ , for } v \neq code(w) \end{aligned}$$

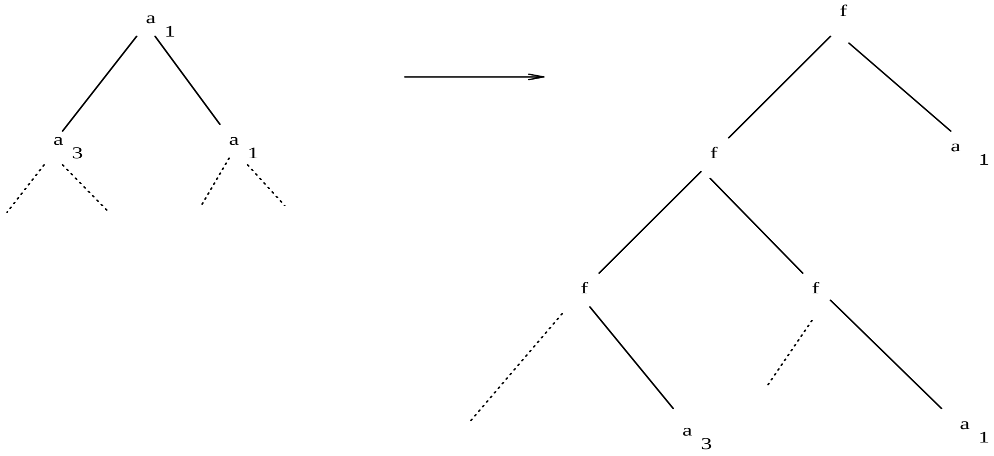

Figure 3: Transformation $t \mapsto t'$.

Let

$$\hat{t} \quad = \quad t'[a_1 \leftarrow b_1, \ldots, a_n \leftarrow b_n]$$

Let, for a set of trees M, $\hat{M} = \{\hat{t} : t \in M\}$.

We claim that the properties of the sets $\hat{M}_n$ and $\hat{N}_n$ are analogous to those of the sets M_n and N_n. That is, for odd n, $\hat{M}_n$ is in $\Sigma^\mu_n(Sig)$ but not in $\Pi^\mu_n(Sig)$; for even n, $\hat{M}_n$ is in $\Pi^\mu_n(Sig)$ but not in $\Sigma^\mu_n(Sig)$; and the similar properties hold for the sets N_n.

Again, we find it convenient to deal with the automata equivalent to the fixed point terms. Construction of the automata justifying the positive part of the claim is easy. The proof of the negative part does not depend on the particular case of M_n or N_n; we shall fix the attention on the set $\hat{M}_{2m+1}$. It is enough to show that $\hat{M}_{2m+1}$ cannot be recognized by a Rabin automaton with index $m+1$ weakened by $\emptyset$ and "by all". Suppose, to the contrary, that $A = \langle Sig, Q, q_0, Tr, Acc\rangle$ is such an automaton. Let $Acc = \{(L_1, U_1), \ldots, (L_{m+1}, U_{m+1})\}$. We shall construct an automaton A' over Sig_{2m+1}, with the same index as A, recognizing M_{2m+1}.

The set of states of A' is $Q \times Q$ and the initial state is $\langle q_0, q_0\rangle$. For $x \in Q$, let $L(A, x)$ denote the set of trees in T_{Sig} for which there exists an accepting x–run of A. The set of transitions Tr' is defined as follows :

> if $x = f(x_1, x_2)$ and $x_1 = f(z, y)$ are the transitions of A, such that moreover $b_i \in L(A, x_2)$, then
> $$\langle x, x'\rangle \;=\; a_i(\langle y, x_1\rangle, \langle z, x_1\rangle)$$
> is a transition of of A', and there are no other transitions.

The acceptance condition of A' is $Acc' = \{(L'_1, U'_1), \ldots, (L'_{m+1}, U'_{m+1})\}$, where, for each $X = L_i$ or $X = U_i$, $X' = (X \times Q) \cup (Q \times X)$. It should be clear that if the index of A is weakened by $\emptyset$ and "by all", so is the index of A'.

Now, given a tree $t \in T_{Sig_{2m+1}}$ and an accepting run of A' on t, it is not difficult to construct an accepting run of A on the tree $\hat{t} \in T_{Sig}$. Conversely, given an accepting run of A on $\hat{t}$, it is not difficult to construct an accepting run of A' on t; we omit the details.

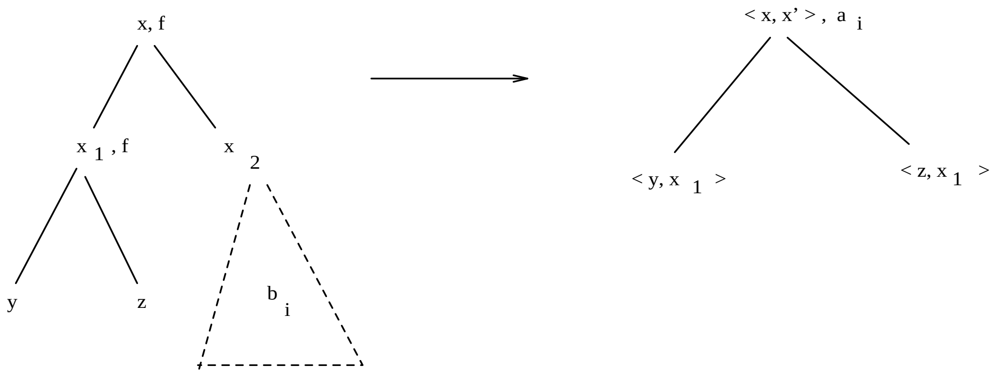

Figure 4: Producing transitions of A'.

Thus $L(A') = M_{2m+1}$, which is a contradiction to Theorem 5.1 (c.f. Claim 5.2, 1). This remark completes the proof. □

For some signatures, however, the hierarchy is finite. The following fact has been shown, in a slightly different formulation, by D. Park [40].

Proposition 5.6 (D. Park) *Let* Sig *be a signature such that the arity of all symbols is at most one. Then*

$$fp(\wp\mathcal{T}_{Sig}) = \Pi_2^{\mu}(\wp\mathcal{T}_{Sig})$$

Proof. The argument follows *via* the equivalence of fixed point terms and Rabin automata from the well–known fact that, for infinite words, Rabin automata can be always simulated by non–deterministic Büchi automata (see [49]). The presence of constants and variables does not require much modifications, so we only briefly sketch the argument for the sake of completeness.

Let $A[\vec{z}]$ be a Rabin automaton with variables, with the acceptance condition $Acc = \{(L_1, U_1), \ldots, (L_n, U_n)\}$. If $n = 0$, this acceptance condition is equivalent to a Büchi condition $F = \emptyset$. Suppose $n > 0$. Since any tree in T_{Sig} has at most one infinite path, the operation defined by A is equivalent to a finite union

$$L(A[\vec{z}]) = \bigcup_{i=1,\ldots,n} L(A_i[\vec{z}])$$

where the automaton A_i differs from A only in that its accepting condition is just $\{(L_i, U_i)\}$. Therefore, without loss of generality we may assume that $m = 1$, say $Acc = \{(L, U)\}$. Now it is an easy exercise to construct a Büchi automaton equivalent to A. □

Remark. The above result is optimal, in the sense that, in general, $\Sigma_2^{\mu}(Sig) \neq \Pi_2^{\mu}(Sig)$. D. Park in [39] exhibits an example that we have mentioned in the introduction : let Sig consists of two unary symbols a and b and let FM ("fair merge") be the set of all trees in T_{Sig} (that is, infinite words over the alphabet $\{a, b\}$), such that both a and b occur infinitely often. It can be shown by a topological argument that FM is in $\Pi_2^{\mu}(Sig)$ but not in $\Sigma_2^{\mu}(Sig)$.

It is not hard to see that the argument we have used for the signature *Sig* above can be carried over to many other signatures. We summarize the above considerations in the following.

Corollary 5.7 *Let* Sig *be a signature containing at least two symbols of which at least one is of an arity* ≥ 2. *Then,* $\Sigma_n^\mu(\mathrm{Sig}) \neq \Pi_n^\mu(\mathrm{Sig})$, *for all* $n < \omega$, *that is, the fixed point hierarchy in the algebra* $\wp\mathcal{T}_{Sig}$ *is infinite. For all other signatures,* $fp(\wp\mathcal{T}_{Sig}) = \Pi_2^\mu(\wp\mathcal{T}_{Sig})$.

Proof. The first part of the claim follows from adaptation of the proof of Theorem 5.5 above; we omit the details. The second part of the claim follows from Proposition 5.6 and an easy observation that, for a signature consisting of *one* symbol, $\Sigma_2^\mu(Sig) = \Pi_2^\mu(Sig)$. $\Box$

Note. One may investigate how does the alternation hierarchy look like in other μ–algebras. The powerset tree algebras are the only examples known to the author in which the hierarchy has been proved infinite. One may however expect that an infinite hierarchy may be also constructed over $\wp(\omega)$ with an appropriate family of arithmetically definable operations, by a diagonalisation method. In [38], the author has considered the hierarchy problem for the powerset algebra of an algebra $\langle \Sigma^\infty, \cdot, \epsilon, \sigma \in \Sigma \rangle$, where $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$ is the set of finite and infinite words over a finite alphabet Σ, and $\cdot$ is a concatenation operation, such that if $u \in \Sigma^\omega$ then $uv = u$, for any $v \in \Sigma^\infty$. It has been proved that the hierarchy is finite, and moreover it collapses on the level $\mathrm{Comp}(\Sigma_1^\mu \cup \Pi_1^\mu)$.

In any *finite* μ–algebra, if we allow the symbols $\top$ and $\bot$ for the greatest and least elements respectively, then any fixed point term is equivalent to a term without fixed point operators. (This follows, e.g., from the iterational characterization of fixed points, Theorem 2.2.) Considering that $\bot = \mu x.x$ and $\top = \nu x.x$, we obtain the collapsing of the hierarchy to the level $\mathrm{Comp}(\Sigma_1^\mu \cup \Pi_1^\mu)$, in all finite μ–algebras.

6 Related formalisms

In this section we compare our fixed point calculus over powerset algebras to some other calculi considered in the literature. Several open problems appear naturally in this context. We also mention a connection between the fixed point calculus and monadic second order logic interpreted over syntactic trees.

6.1 The fixed point calculus with intersection

It is natural to extend the syntax of our fixed point calculus by a binary operator $\wedge$ interpreted as the greatest lower bound in the universe of a μ–algebra. In a powerset algebra, this operator is just as the set–theoretical intersection.

The fixed point calculus with intersection interpreted in the powerset algebra of syntactic trees has been considered by Arnold and Niwiński [2, 3]. Clearly, the extension is proper, as, e.g., the term $x \wedge y$ is not equivalent to any fixed point term without intersection. However, the extended calculus does not define more tree languages; that is, for any *closed* fixed point term with intersection τ, there exists a fixed point term without intersection τ' such that τ and τ' define the same set of trees when interpreted in the powerset tree algebra. The existence of such τ' can be inferred from known results : an embedding (easy) of the fixed point calculus with intersection into the monadic second order logic (c.f. Section 6.3 below), a characterization of this logic by Rabin automata proved by Rabin [41], and, finally, our characterization of Rabin automata by the fixed point calculus without intersection (Theorems 3.2 and 3.3). However, we should expect that the complexity of a translation from τ to τ' will be high; in particular, the number of alternations of μ and ν may, in course of this translation, increase arbitrarily. The last follows from an observation that our tree languaues exhibiting the infinity of fixed point hierarchy for terms without intersection (Section 5) are all the complements of some Büchi languages[9] and therefore can be defined by terms with intersection of the level Σ_2^μ.

It can be noted [2] that the automata counterpart of the fixed point terms with intersection is provided by *alternating automata* introduced by Muller and Schupp [32]. Therefore, the elimination of intersection in the fixed point calculus amounts to the reduction of alternating automata to nondeterministic automata. (A purely automata–theoretic proof of this last fact is given by Muller and Schupp [10] [33].) We belive that an analogous result can be proved for the fixed point calculus interpreted over the class of all powerset algebras; Janin and Walukiewicz [21] have recently shown a similar fact for the modal μ–calculus interpreted over transition systems.

In [2], the hierarchy problem for the fixed point calculus with intersection is also considered. It is proved that the tree languages definable on the level Π_2^μ of this hierarchy coincide with those definable on the analogous level in the fixed point calculus without intersection (as we have remarked above, this level can be also characterized by Büchi automata). On the other hand, it is observed there that the levels Σ_2^μ of both hierarchies are different and that, in general, $\Pi_2^\mu \neq \Sigma_2^\mu$ also in the presence of intersection. The last was the most that we were able to say about the hierarchy in that case; in particular, the question whether it is infinite has remained open. We note that Thilo Hafer has shown in his dissertation [19] that Boolean combinations of tree languages recognizable by Büchi automata (that coincide with the Boolean combinations of tree languages definable on the levels Π_2^μ and Σ_2^μ) do not exhaust all the languages definable by Rabin automata. Recently, G. Lenzi [27] and J. Bradfield [4] have presented two different proofs of the fact that a fixed point hierarchy for the modal μ–calculus (see below) is infinite. This result implies that the height of the hierarchy cannot be uniformly bounded for all powesrest algebras of trees, however it does not yet follow that there exists a single signature *Sig* such that the fixed point hierarchy in $\wp\mathcal{T}_{Sig\wedge}$ is infinite.

Concerning the signatures where the arity of function symbols is at most 1 (and so the trees can be viewed as possibly infinite words), it is shown in [3] that the hierarchy of tree languages definable by fixed point terms with intersection collapses on the level $\mathrm{Comp}(\Pi_1^\mu \cup \Sigma_1^\mu)$, thus earlier than in the case without intersection (Π_2^μ).

6.2 Modal mu–calculus

Modal μ–calculus has been defined by Kozen [23] as an extension of the propositional modal logic, usually with many modalities, by the least fixed point operator; the greatest

[9] This observation is due to A. Arnold.

[10] The paper is dedicated to the memory of Ahmed Saoudi.

fixed point is definable by the duality law : $\nu X.p(X) = \neg\mu X.\neg p(\neg X)$. The modal μ–calculus can be presented as a fragment of our fixed point calculus as follows. Let Act and $Prop$ be finite sets of symbols called actions and propositions, respectively. We consider a signature $Sig_{modal} = Prop \cup\{\bar{p} : p \in Prop\} \cup\{\langle a\rangle : a \in Act\} \cup\{[a] : a \in Act\}$, where the symbols p, $\bar{p} \in Prop$, are considered as 0–ary, and the symbols $\langle a\rangle$ and $[a]$ as unary function symbols. The formulas of the modal μ–calculus are fixed point terms over the signature $Sig_{modal,\vee\wedge}$ (c.f. Section 1.2). Classically, models for this calculus are so–called *Kripke structures* of the form $M = \langle S^M, \{p^M \subseteq S^M : p \in Prop\}, \{a^M \subseteq S^M \times S^M : a \in Act\}\rangle$, where S^M is an underlying set of states (or worlds), and the p^M's and a^M's are interpretations of propositions and actions, respectively. According to the classical interpretation of the modal operators $\langle a\rangle$ and $[a]$, we can associate with such a model M a μ–algebra, say $\mathcal{M}$, over signature $Sig_{modal,\vee\wedge}$, with the universe $\wp(S^M)$ completely ordered by subset ordering,

$$\begin{array}{rcl}\mathcal{M} & = & \langle \wp(S^M), \{p^{\mathcal{M}} : p \in Prop\} \cup \{\bar{p}^{\mathcal{M}} : p \in Prop\} \\ & & \cup\{\langle a\rangle^{\mathcal{M}} : a \in Act\} \cup \{[a]^{\mathcal{M}} : a \in Act\} \cup \{\vee^{\mathcal{M}}, \wedge^{\mathcal{M}}\}\rangle\end{array}$$

where $\vee^{\mathcal{M}}$ and $\wedge^{\mathcal{M}}$ are set–theoretical union and intersection respectively, $p^{\mathcal{M}} = p^M$ and $\bar{p}^{\mathcal{M}} = S^M - p^M$, for $p \in Prop$, and, for $L \subseteq S^M$,

$$\begin{array}{rcl}\langle a\rangle^{\mathcal{M}}(L) & = & \{s \in S^M : (\exists s' \in S^M)\, \langle s, s'\rangle \in a^M \wedge s' \in L\} \\ {[a]}^{\mathcal{M}}(L) & = & \{s \in S^M : (\forall s' \in S^M)\, \langle s, s'\rangle \in a^M \Rightarrow s' \in L\}\end{array}$$

Then the classical interpretation of the formulas of the modal μ–calculus [23] coincide with our interpretation of fixed point terms. We note however that a μ–algebra $\mathcal{M}$ cannot in general be easily identified with any powerset algebra $\wp\mathcal{B}$ (the difficulty stems from the *a priori* unbounded arity of $[a]$).

Conversely, for an arbitrary signature Sig, we can consider a vocabulary of the modal μ–calculus $Prop = \{p_f : f \in Sig\}$, $Act = \{d_i : 1 \leq i \leq k\}$, where k is the maximum of arities of the symbols in Sig. In particular, a tree $t \in T_{Sig}$ can be considered as a Kripke structure over this vocabulary, say $\tilde{t}$, with $S^{\tilde{t}} = \operatorname{dom} t$, $p_f^{\tilde{t}} = \{w \in \operatorname{dom} t : t(w) = f\}$, and $d_i^{\tilde{t}} = \{\langle w, wi\rangle : w, wi \in \operatorname{dom} t\}$. Let Sig_{modal} be the signature constructed as above on the basis of sets $Prop$ and Act. Then we can embed fixed point terms over signature $Sig_{\vee,\wedge}$ into the modal μ–calculus over this vocabulary by the following translation :

$$\begin{array}{rcl}e(x) & = & x \\ e(\tau_1 \vee \tau_2) & = & e(\tau_1) \vee e(\tau_2) \\ e(\tau_1 \wedge \tau_2) & = & e(\tau_1) \wedge e(\tau_2) \\ e(f(\tau_1, \ldots, \tau_{\rho(f)})) & = & p_f \wedge \langle d_1\rangle e(\tau_1) \wedge \ldots \wedge \langle d_{\rho(f)}\rangle e(\tau_{\rho(f)}) \\ e(\mu x.\tau) & = & \mu x.e(\tau) \\ e(\nu x.\tau) & = & \nu x.e(\tau)\end{array}$$

Recall that a tree $t \in T_{Sig}$ can be also viewed as a semi–algebra over signature Sig, $\mathbf{t}$ (Section 1.2). Then it is easy to verify that, for each fixed point term τ, $\tau^{\mathbf{t}} = e(\tau)^{\tilde{t}}$. Considering the internalization property (Proposition 4.2), we can say that our fixed point calculus interpreted over powerset algebra of trees can be embedded into the modal μ–calculus.

Note. The above construction does not directly carry over to arbitrary semi–algebras, since we can meet the following problem. Suppose that in an algebra we have $c \doteq f(a,b)$ and $c \doteq f(b,a)$, but not $c \doteq f(a,a)$. Then it seems to be no obvious way to construct a Kripke structure over the same domain, such that the fixed point terms could be translated into equivalent modal formulas (unless the number of modalities excessively increases). Therefore we do not believe that our fixed point calculus over arbitrary powerset algebras is trivially subsumed by the modal μ–calculus.

The formulas of the modal μ–calculus resulting from the aforementioned translation e have a special form, since they use the conjunction in a restricted way. Such formulas do not exhaust the class of all modal formulas, even if one restricts the class of models to the syntactic trees. However, similarly as for the fixed point calculus with intersection discussed above, it can be shown that any closed formula of the modal μ–calculus is equivalent to some $e(\tau)$ over syntactic trees. Interestingly, Janin and Walukiewicz [21] present a class of what they call *disjunctive formulas*, which can be viewed as a generalization of the formulas $e(\tau)$, and prove that any formula of the modal mu–calculus is equivalent to some disjunctive formula.

6.3 Monadic second order logic

A connection between recognizability by finite state automata and definability in monadic second order logic has been established by Büchi [5], who also extended the concept of finite state recognizability to *infinite* words, in order to prove decidability of the monadic second order theory of ω with successor operation (only). Rabin [41] used the concept of automata on infinite *trees* to prove decidability of the monadic second order theory of full k–ary tree (for arbitrary $k \leq \omega$). The structure in consideration is the (unlabeled) k–ary tree with k successor operations. The lapelled trees appear in an intermediate stage of the proof, because a valuation of monadic second order variables, say $Z_1 \mapsto K_1, \ldots, Z_m \mapsto K_m$ (where $K_1, \ldots, K_m$ are sets of nodes) can be identified with a labeling of the nodes of the tree in an alphabet $\{0,1\}^m$ (such that the the i-th component of the label of w is 1 iff $w \in K_i$); the key fact is that a set of trees over this alphabet is recognizable by a Rabin automaton iff the corresponding set of valuations is definable by a monadic second order formula. Adaptation of this last characterization to syntactic trees is straightforward; we shall note its prolongation to the fixed point calculus.

Let $t \in T_{Sig}$ and let $\tilde{t} = \langle \operatorname{dom} t, \{p_f^{\tilde{t}} : f \in Sig\}, \{d_i^{\tilde{t}} : 1 \leq i \leq k\} \rangle$ be the Kripke structure as defined above in the context of modal μ–calculus. It can be viewed as a logical structure over a vocabulary consisting of monadic symbols p_f, for $f \in Sig$, and binary symbols d_i, $i \leq k$, where k is the maximum of arities of the symbols in Sig.

The formulas of monadic second order logic use two kinds of variables : *individual variables* $x_0, x_1, \ldots$, ranging over elements of $\operatorname{dom} t$, and *set variables* $X_0, X_1, \ldots$, ranging over subsets of $\operatorname{dom} t$. Atomic formulas are $x_i = x_j$, $p_f(x_i)$, $d_m(x_i, x_j)$, $X_i(x_j)$. The other formulas are built using propositional connectives $\vee, \wedge, \neg, \Rightarrow, \Leftrightarrow$, and quantifiers $\forall, \exists$ ranging over both kinds of variables. A formula without free variables is called a *sentence*. The *satisfaction* of formulas in $\tilde{t}$ is defined in the usual way.

For a sentence φ, let $\operatorname{Mod}(\varphi)$ denote the set of all $t \in T_{Sig}$, such that $\tilde{t}$ satisfies φ. Then, from the Rabin's characterization of monadic second order logic and our characterization of automata, we can infer the following.

Theorem 6.1 *For any fixed point term* τ *over* Sig, *one can construct a monadic second order sentence* φ_τ, *such that* $\tau^{\wp\mathcal{T}_{Sig}} = Mod(\varphi_\tau)$.

Conversely, for any monadic second order sentence φ, *one can construct a fixed point term* τ_φ, *such that* $Mod(\varphi) = \tau_\varphi^{\wp\mathcal{T}_{Sig}}$.

The first part of the above theorem can be proved directly, without using automata, but using the internalization property (Proposition 4.2), as the least and greatest fixed points in $\wp(\mathrm{dom}\, t)$ can be easily expressed in monadic second order logic (a proof of this fact for the fixed point calculus with intersection can be found in [3]).

We note that the two equivalent formalisms considered in above theorem are extremely distant in succinctness. Indeed, the complexity of the emptiness problem "$\tau_\varphi^{\wp\mathcal{T}_{Sig}} \neq \emptyset$?" is in *PTIME* (Corollary 4.5), while the complexity of the analogous problem "$\mathrm{Mod}(\varphi) \neq \emptyset$?" is known to be non-elementary [14].

Acknowledgments. I wish to thank Jerzy Tiuryn for introducing me to the theory of fixed points. I am greatful to Andrzej W. Mostowski, André Arnold, Bruno Courcelle and Wolfgang Thomas for enlightening discussions and for encouragement. I am also indebted to the referees for corrections and helpful comments, and to Igor Walukiewicz for his critical remarks on the preliminary version of this paper.

References

[1] A. Arnold and M. Nivat, Non deterministic recursive programs, in : Proc. FCT'77 Lecture Notes in Computer Science, Vol. 56 (Springer, Berlin, 1977) 12–21.

[2] A. Arnold and D. Niwiński, Fixed point characterization of Büchi automata on infinite trees, *J. Inf. Process. Cybern. EIK* **26** (1990) 453–461.

[3] A. Arnold and D. Niwiński, Fixed point characterization of weak monadic logic definable sets of trees, in : M. Nivat and A. Podelski, eds., Tree Automata and Languages (Elsevier, Amsterdam, 1992) 159–188.

[4] J. C. Bradfield, The modal mu-calculus alternation hierarchy is strict, in : Proc. CONCUR'96, Lecture Notes in Computer Science, Vol. 1119 (Springer, Berlin, 1996) 233–246.

[5] J. R. Büchi, On a decision method in restricted second order arithmetic, in : E. Nagl, ed., *Logic, Methodology, and Philosophy of Science*, Stanford Univ. Press (1960) 1–11.

[6] B. Courcelle, Equivalences and transformations of regular systems. Applications to recursive program schemes and grammars, *Theoret. Comput. Sci.* **42** (1986) 1–122.

[7] E.A. Emerson, The Role of Büchi's Automata in Computing Science, *in :* S. MacLane and D. Siefkes, eds., *The Collected Works of J. R. Büchi* (Springer, Berlin, 1990).

[8] E.A. Emerson and E.M. Clark, Characterizing correctness properties of parallel programs using fixpoints, in : *Proc. ICALP'80*, Lecture Notes in Computer Science, Vol. 85 (Springer, Berlin, 1980) 169-181.

[9] E.A. Emerson and C.S.Jutla, The Complexity of Tree Automata and Logics of Programs, in : *Proc. 29th IEEE Symp. on Foundations of Computer Science* (1988) 328-337.

[10] E.A. Emerson and C.S. Jutla, On Simultaneously Determinizing and Complementing ω-Automata, in : *Proc. 4th IEEE Symp. on Logic in Comput. Sci.* (1989) 333-342.

[11] E.A. Emerson, Temporal and Modal Logic, in : J. van Leeuven, ed., *Handbook of Theoretical Computer Science. vol.B* (Elsevier, Amsterdam, 1990) 995-1072.

[12] E.A. Emerson and C. Lei, Efficient model checking in fragments of the propositional mu–calculus, in : *Proc. 1st IEEE Symp. on Logic in Comput. Sci.* (1986) 267-278.

[13] E.A. Emerson and C.S. Jutla, Tree Automata, Mu-Calculus and Determinacy, in : *Proc. 32nd IEEE Symp. on Foundations of Computer Science* (1991) 368-377.

[14] J. Ferrante and C.W. Rackoff, The computational complexity of logical theories, Lecture Notes in Mathematics Vol. 718 (Springer, Berlin, 1979).

[15] L. Flon and N. Suzuki, Consistent and Complete Proof Rules for the Total Correctness of Parallel Programs, in : *Proc. 19th IEEE Symp. on Foundations of Computer Science* (1978).

[16] I. Guessarian, Algebraic Semantics, Lecture Notes in Computer Science, Vol. 99 (Springer, Berlin, 1981).

[17] Y.Gurevich and L. Harrington, Trees, Automata and Games, in : *Proc. 14th ACM Symp. on Theory of Computation* (1982) 60–65.

[18] G. Grätzer, Universal Algebra (D. Van Nostrand Co. Inc., Princeton, 1968).

[19] T. Hafer, Automaten und Logiken über unendlichen Bäumen, Ph.D. Thesis, RWTH Aachen, 1989.

[20] R.Hossley and C.Rackoff, The Emptiness Problem For Automata on Infinite Trees, in : *Proc. 13th IEEE Symp. Switching and Automata Theory* (1972) 121-124.

[21] D.Janin and I.Walukiewicz, Automata for the μ–calculus and related results, in : *Proc. MFCS'95*, Lecture Notes in Computer Science, Vol. 969 (Springer, Berlin, 1995) 552-562.

[22] B. Knaster, Un théorème sur les fonctions des ensembles, *Ann. Soc. Polon. Math* **6** (1928) 133–134.

[23] D.Kozen, Results on the propositional mu-calculus, *Theoret. Comput. Sci.* **27** (1983) 333-354.

[24] D.Kozen and R.J.Parikh, A decision procedure for the propositional mu-calculus, in : *Second Workshop on Logics of Programs*, Lecture Notes in Computer Science, Vol. 164 (Springer, Berlin, 1983) 313-325.

[25] D.Kozen and J.Tiuryn, Logics of programs, in : J. van Leeuven, ed., *Handbook of Theoretical Computer Science. vol.B* (Elsevier, Amsterdam, 1990) 789-840.

[26] K. Kuratowski and A. Mostowski, Set theory (North Holland, Amsterdam–Warsaw, 1976).

[27] G. Lenzi, A hierarchy theorem for the μ–calculus, in : *Proc. ICALP'96*, Lecture Notes in Computer Science, Vol. 1099 (Springer, Berlin, 1996) 87-109.

[28] A.W. Mostowski, Regular Expressions for Infinite Trees and a Standard Form of Automata, in : *Computation Theory*, Lecture Notes in Computer Science, Vol. 208 (Springer, Berlin, 1985) 157–168.

[29] A.W. Mostowski, Games with forbidden positions, Technical Report 78, University of Gdańsk, 1991.

[30] R. McNaughton, Infinite games played on finite graphs, *Ann. Pure Appl. Logic* **65** (1993) 149–184.

[31] D.E. Muller, Infinite sequences and finite machines, in. : *Proc. 4th Annual IEEE Symposium on Switching Circuit Theory and Logical Design* (1963) 3–16.

[32] D.E. Muller and P. Schupp, Alternating automata on infinite trees, *Theoret. Comput. Sci.* **54** (1987) 267–276.

[33] D.E. Muller and P. Schupp, Simulating alternating tree automata by nondeterministic automata : new results and new proofs of the theorems of Rabin, McNaughton and Safra, *Theoret. Comput. Sci.* **141** (1995) 69-107.

[34] D. Niwiński, Fixed–Point Characterization of Context–Free ∞–languages, *Inform. and Contr.* **61** (1984) 247–276.

[35] D. Niwiński, Equational μ–calculus, in : *Computation Theory*, Lecture Notes in Computer Science, Vol. 208 (Springer, Berlin, 1985) 169–176.

[36] D. Niwiński, On fixed point clones. In L. Kott, editor, *Proc. 13th ICALP'86*, Lecture Notes in Computer Science, Vol. 226 (Springer, Berlin, 1986) 464–473.

[37] D. Niwiński, Fixed points vs infinite generation, in : *Proc. 3rd IEEE Symp. on Logic in Comput. Sci.* (1988) 402–409.

[38] D. Niwinski, Hierarchy of objects definable in the fixed point calculus, Ph.D. Thesis, University of Warsaw, 1987 (in Polish).

[39] D.M.R.Park, On the semantics of fair parallelism, in : *Abstract software specification*, Lecture Notes in Computer Science, Vol. 86 (Springer, Berlin, 1980) 504–526.

[40] D.M.R.Park, Fixpoint Induction and Proof of Program Semantics, in : *Machine Intelligence, Vol.5* (Edinburgh University Press, 1969) 59-78.

[41] M.O. Rabin, Decidability of second-order theories and automata on infinite trees, *Trans.Amer.Soc.* **141** (1969) 1-35.

[42] M.O. Rabin, Weakly definable relations and special automata, in : Y. Bar-Hillel, ed., *Mathematical Logic in Foundations of Set Theory* (1970) 1-23.

[43] M.O. Rabin, Automata on infinite objects and Church 's problem, *Amer. Math. Soc.*, (Providence RI 1972) 1-22.

[44] S. Safra, On the Complexity of ω-Automata, in : *Proc. 29th IEEE Symp. on Foundations of Computer Science* (1988) 319-327.

[45] C.P.Stirling and D.J.Walker, Local model checking in the modal mu-calculus, in : *Proc. International Joint Conference in Theory and Practice of Software Development*, Lecture Notes in Computer Science, Vol. 351 (Springer, Berlin, 1989) 369-382.

[46] R.S. Streett and E.A. Emerson, An automata theoretic procedure for the propositional mu-calculus, *Information and Computation* **81** (1989) 249-264.

[47] M. Takahashi, The greatest fixed points and rational omega–tree languages, *Theoret. Comput. Sci.* **44** (1986) 259-274.

[48] A. Tarski, A lattice theoretical fixpoint theorem and its applications. *Pacific J. of Math.* **5** (1955) 285–309.

[49] W. Thomas, Automata on infinite objects, in : J. van Leeuven, ed., *Handbook of Theoretical Computer Science. vol.B* (Elsevier, Amsterdam, 1990) 133-191.

[50] W. Thomas, On the synthesis of strategies in infinite games, in : *Proc. STACS'95*, Lecture Notes in Computer Science Vol. 900 (Springer, Berlin, 1995) 1–13.

[51] M.Y. Vardi and P. Wolper, Automata–theoretic techniques for modal logic of programs, *J. Comput. System Sci.* **18** (1986) 183-221.

[52] K. Wagner, Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen, *J. Inf. Process. Cybern. EIK* **13** (1977) 473-487.

[53] K. Wagner, On ω–regular sets, *Inform. and Contr.* **43** (1979) 123-177.

[54] I. Walukiewicz, Completeness of Kozen's axiomatisation of the propositional μ–calculus, in : *Proc. 10th IEEE Symp. on Logic in Comput. Sci.* (1995) 14-24.

Damian Niwiński
Institute of Informatics
University of Warsaw
ul. Banacha 2, 02 097 Warszawa
Poland
e–mail : `niwinski@mimuw.edu.pl`