Basic Research in Computer Science

BRICS RS-98-53 J. C. Bradfield: Fixpoint Alternation: Arithmetic, Transition Systems, and the Binary Tree

Fixpoint Alternation: Arithmetic, Transition Systems, and the Binary Tree

Julian C. Bradfield

BRICS Report Series **RS-98-53**

ISSN 0909-0878 **December 1998**

See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/98/53/`

Fixpoint alternation: arithmetic, transition systems, and the binary tree[1]

J. C. Bradfield

BRICS,[2] Department of Computer Science, Aarhus University, Bygning 540, Ny Munkegade, 8000 Århus C, Denmark
and
LFCS,[3] Division of Informatics, University of Edinburgh, Edinburgh, United Kingdom, EH9 3JZ (email: jcb@dcs.ed.ac.uk)

Abstract: We provide an elementary proof of the fixpoint alternation hierarchy in arithmetic, which in turn allows us to simplify the proof of the modal mu-calculus alternation hierarchy. We further show that the alternation hierarchy on the binary tree is strict, resolving a problem of Niwiński.

1 Introduction

The modal mu-calculus, or Hennessy–Milner logic with fixpoints, is a popular logic for expressing temporal properties of systems. It was first studied by Kozen in [Koz83], and since then there has been much work on both theoretical and practical aspects of the logic. The feature of the logic that gives it both its simplicity and its power is that it is possible to have mutually dependent minimal and maximal fixpoint operators. This makes it simple, as the fixpoints are the only non-first-order operators, and powerful, as by such nesting one can express complex properties such as 'infinitely often' and fairness. A measure of the complexity

[1] This is a manuscript submitted for publication. It extends and revises [Bra98], a preprint of which appeared as LFCS technical report ECS–LFCS–98–385.
[2] Danish National Research Foundation Centre for Basic Research in Computer Science
[3] Laboratory for Foundations of Computer Science

of a formula is the *alternation depth*, that is, the number of alternating blocks of minimal/maximal fixpoints. Formulae of alternation depth higher than 2 are notoriously hard to understand, and in practice one rarely produces them—not least because they are so hard to understand. For some years, it was not even known whether formulae of high alternation depth were necessary, that is, whether the *alternation hierarchy* was indeed a strict hierarchy of expressive power—a problem with several interesting ramifications, as well as its intrinsic interest. In 1996 the strictness of the hierarchy was established by the present author [Bra97], and independently by Lenzi [Len96]. The proof technique in [Bra97] relied on the existence of a similar fixpoint alternation hierarchy in arithmetic with fixpoints (mu-arithmetic). Mu-arithmetic appears, somewhat surprisingly, not to have been studied in the recursion-theoretic literature, apart from the deep and technical recursion-theoretic study of mu-arithmetic by Robert Lubarsky [Lub93], which incidentally implies the hierarchy. Thus the proof of [Bra97] was not self-contained; furthermore, it was apparently not feasible to exhibit examples of strict alternation depth n formulae, as the strict mu-arithmetic formulae of [Lub93] are not constructible with any reasonable amounts of paper, ink and patience—there is only a high level description of the complex coding required, which not only establishes the hierarchy but also gives a precise characterization in terms of large admissible ordinals.

Another question is whether the alternation hierarchy remains strict on trees of bounded branching degree, in particular the binary tree. This is closely related to another long-standing alternation problem arising from Damian Niwiński's study of fixpoint algebras over trees. In his papers [Niw86,Niw97] he has a fixpoint logic for such algebras in which formulae are built from n-ary function symbols, disjunction, and least and greatest fixpoint operators. The structures are infinite trees such that each node with n children is labelled with some n-ary function symbol; a node satisfies the formula $f(\phi_1, \ldots, \phi_n)$ if the node is labelled by f and the ith child satisfies ϕ_i; the fixpoints are taken over sets of nodes; a tree satisfies a formula if its root does. Niwiński established a number of results about such algebras, including intimate and now well-known relationships to automata theory. One result in particular concerns us here: he established a strict hierarchy of tree languages definable according to the number of alternating fixpoint quantifiers in the formula. In fact this hierarchy

is closely connected to the hierarchy of Rabin indices in Rabin automata languages. However, in this study, as we have mentioned, the only boolean operator was union, and *not* intersection. Although intersection is easily added as a function symbol, many of the results rely on it not being a primitive of the language; and in particular, the alternation hierarchy theorem is not established for the languages with intersection. This problem has remained open.

In this paper we address these problems. The first result is an elementary proof of the alternation hierarchy in mu-arithmetic. This proof uses the standard technique for recursion-theoretic hierarchies; thus we remove the reliance on [Lub93]. Furthermore, the proof constructs very simple examples of strict level n formulae of mu-arithmetic; and by using a simplified version of the techniques of [Bra97], we are able to construct even simpler examples of strict level n modal mu-calculus formulae. These examples are of just the form that one expects, if one is a modal mu-calculus hacker. In addition, we can also show that the formulae defining the existence of a winning strategy in a parity game are examples of strict formulae—indeed, a referee has observed that this can be shown already from [Bra97]. We then show how the proof can be carried through for the case of bounded branching degree systems, and then for the case of Niwiński's logic, so resolving the problem left open in [Niw86].

The material of sections 3–5 was first presented at STACS '98 [Bra98]; sections 6–7 were presented at FICS in Brno in 1998 [Bra98a].

2 Preliminaries

2.1 Modal mu-calculus

We assume some familiarity with the modal mu-calculus, so in this section we give brief definitions to establish notations and conventions. Expository material on the modal mu-calculus may be found in [Bra91,Sti91].

The modal mu-calculus, with respect to some countable set $\mathscr{L}$ of *labels*, has formulae Φ defined inductively thus: variables Z and the boolean constants tt, ff are formulae; if Φ_1 and Φ_2 are formulae, so are $\Phi_1 \vee \Phi_2$ and $\Phi_1 \wedge \Phi_2$; if Φ is a formula and l a label, then $[l]\Phi$ and $\langle l \rangle \Phi$ are formulae; and if Φ is a formula and Z a variable, then $\mu Z.\, \Phi$ and $\nu Z.\, \Phi$ are formulae.

Note that we adopt the convention that the scope of the binding operators μ and ν extends as far as possible. For consistency, we also apply

this convention to the $\forall$ and $\exists$ of first-order logic, writing $\forall x.\,(\exists y.\,P) \vee Q$ rather than the logicians' traditional $\forall x\,[\exists y\,[P] \vee Q]$.

Observe that negation is not in the language, but any closed mu-formula can be negated by using the usual De Morgan dualities—μ and ν are dual by $\neg\mu Z.\,\Phi(Z) = \nu Z.\,\neg\Phi(\neg Z)$. Where necessary, we shall assume that free variables can be negated just by adjusting the valuation. We shall use $\Rightarrow$ etc. freely, though we must ensure that bound variables only occur positively.

We use the symbol $\mu\nu$ to mean 'μ or ν as appropriate'.

Given a labelled transition system $\mathscr{T} = (\mathscr{S}, \mathscr{L}, \longrightarrow)$, where $\mathscr{S}$ is a set of states, $\mathscr{L}$ a set of labels, and $\longrightarrow \subseteq \mathscr{S} \times \mathscr{L} \times \mathscr{S}$ is the transition relation (we write $s \xrightarrow{l} s'$), and given also a valuation $\mathscr{V}$ assigning subsets of $\mathscr{S}$ to variables, the denotation $\|\Phi\|^{\mathscr{T}}_{\mathscr{V}} \subseteq \mathscr{S}$ of a mu-calculus formula Φ is defined in the obvious way for the variables and booleans, for the modalities by

$$\|[l]\Phi\|^{\mathscr{T}}_{\mathscr{V}} = \{\, s \mid \forall s'.\, s \xrightarrow{l} s' \;\Rightarrow\; s' \in \|\Phi\|^{\mathscr{T}}_{\mathscr{V}} \,\}$$

$$\|\langle l\rangle\Phi\|^{\mathscr{T}}_{\mathscr{V}} = \{\, s \mid \exists s'.\, s \xrightarrow{l} s' \wedge s' \in \|\Phi\|^{\mathscr{T}}_{\mathscr{V}} \,\},$$

and for the fixpoints by

$$\|\mu Z.\,\Phi\|^{\mathscr{T}}_{\mathscr{V}} = \bigcap\{\, S \subseteq \mathscr{S} \mid \|\Phi\|^{\mathscr{T}}_{\mathscr{V}[Z:=S]} \subseteq S \,\}$$

$$\|\nu Z.\,\Phi\|^{\mathscr{T}}_{\mathscr{V}} = \bigcup\{\, S \subseteq \mathscr{S} \mid S \subseteq \|\Phi\|^{\mathscr{T}}_{\mathscr{V}[Z:=S]} \,\}.$$

It is often useful to think of μZ and νZ as meaning respectively finite and infinite looping from Z back to μZ (νZ) as one 'follows a path of the system through the formula'. Examples of properties expressible by the mu-calculus are 'always (on a-paths) P', as $\nu Z.\,P \wedge [a]Z$, 'eventually (on a-paths) P', as $\mu Z.\,P \vee \langle a\rangle Z$, and 'there is an $\{a,b\}$-path along which b happens infinitely often', as $\nu Y.\,\mu Z.\,\langle b\rangle Y \vee \langle a\rangle Z$. (For the latter, we can loop around Y for ever, but each internal loop round Z must terminate.)

There are several notions of alternation. The naive notion is simply to count syntactic alternations of μ and ν, resulting in the following definition: A formula Φ is said to be in the classes Σ^{μ}_{0} and Π^{μ}_{0} iff it contains no fixpoint operators ('S' for 'simple' or 'syntactic'). The class Σ^{μ}_{n+1} is the least class containing $\Sigma^{\mu}_{n} \cup \Pi^{\mu}_{n}$ and closed under the following operations: (i) application of the boolean and modal combinators; (ii) the formation of

$\mu Z.\Phi$, where $\Phi \in \Sigma^\mu_{n+1}$. Dually, to form the class Π^μ_{n+1}, take $\Sigma^\mu_n \cup \Pi^\mu_n$, and close under (i) boolean and modal combinators, (ii) $\nu Z.\Phi$, for $\Phi \in \Pi^\mu_{n+1}$. Thus the examples above are in Π^μ_1, Σ^μ_1, and Π^μ_2 (but not Σ^μ_2) respectively. We shall say a formula is *strict* Σ^μ_n if it is in $\Sigma^\mu_n - \Pi^\mu_n$.

For the modal mu-calculus, it is usual to define stronger notions of alternation [EmL86,Niw86], which capture the true interdependency of alternating fixpoints, rather than just their syntactic position. In [Bra97], the analysis is carried out for the strongest notion, that of [Niw86], giving the classes called $\Sigma^{N\mu}_n$ in [Bra97], as well as for the simple notion. In this paper, we shall not worry about the distinction, as the arguments apply whichever notion is used. Hence we shall just write Σ^μ_n.

2.2 The arithmetic mu-calculus

In [Lub93] Robert Lubarsky studies the logic given by adding fixpoint constructors to first-order arithmetic. Precisely, the logic ('mu-arithmetic' for short) has as basic symbols the following: function symbols f, g, h; predicate symbols P, Q, R; first-order variables x, y, z; set variables X, Y, Z; and the symbols $\vee, \wedge, \exists, \forall, \mu, \nu, \neg, \in$. As with the modal mu-calculus, $\neg$ can be pushed inwards to apply only to atomic formulae, by De Morgan duality.

The language has expressions of three kinds, individual terms, set terms, and formulae. The individual terms comprise the usual terms of first-order logic. The set terms comprise set variables and expressions $\mu(x, X).\,\phi$ and $\nu(x, X).\,\phi$, where X occurs positively in ϕ. Here μ binds both an individual variable and a set variable; henceforth we shall write just $\mu X.\,\phi$, and assume that the individual variable is the lower-case of the set variable. The formulae are built by the usual first-order construction, together with the rule that if τ is an individual term and Ξ is a set term, then $\tau \in \Xi$ is a formula.

This language is interpreted over the structure $\mathbb{N}$ of first-order arithmetic with all recursive functions and predicates—in particular, let $\langle -, - \rangle$, $(-)_0$ and $(-)_1$ be standard pairing and unpairing functions. The semantics of the first-order connectives is as usual; $\tau \in \Xi$ is interpreted naturally; and the set term $\mu X.\,\phi(x, X)$ is interpreted as the least fixpoint of the functional $\mathbf{X} \mapsto \{\, m \in \mathbb{N} \mid \phi(m, \mathbf{X}) \,\}$ (where $\mathbf{X} \subseteq \mathbb{N}$).

The simplest examples of mu-arithmetic just use least fixpoints to represent an inductive definition. For example, $\mu X.\, x = 0 \vee (x > 1 \wedge (x-2) \in X)$ is the set of even numbers. Of course, the even numbers are also

the complement of the odd numbers: the odd numbers are defined by $\mu X.\, x = 1 \vee (x > 1 \wedge (x-2) \in X)$, so by negating we can express the even numbers as a maximal fixpoint $\nu X.\, x \neq 1 \wedge (x > 1 \Rightarrow (x-2) \in X)$. To produce natural examples involving alternating fixpoints is rather difficult, since even one induction is already very powerful, and most natural mathematical objects are simple.

One can define the syntactic alternation classes for arithmetic just as for the modal mu-calculus: First-order formulae are Σ_0^μ and Π_0^μ, as are set variables. The Σ_{n+1}^μ formulae and set terms are formed from the $\Sigma_n^\mu \cup \Pi_n^\mu$ formulae and set terms by closing under (i) the first-order connectives and (ii) forming $\mu X.\, \phi$ for $\phi \in \Sigma_{n+1}^\mu$.

A crucial lemma is the following:

Lemma 1 [Lub93,Bra97] *A Σ_n^μ formula of mu-arithmetic can be put into a normal form of the following shape:*

$$\tau_n \in \mu X_n.\, \tau_{n-1} \in \nu X_{n-1}.\, \tau_{n-2} \in \mu X_{n-2}.\, \ldots \tau_1 \in \mu\nu X_1.\, \phi$$

where ϕ is first-order—that is, a string of alternating fixpoint quantifiers, and a first-order body. □

(See [Bra97] for detailed definitions and proof.)

The analysis of [Lub93] provides the following

Theorem 2 [Lub93] *The hierarchy of the sets of integers definable by Σ_n^μ formulae of the arithmetic mu-calculus is a strict hierarchy.* □

2.3 Summary of [Bra97]

Our results here require the results and proof techniques of [Bra97], so we now give a summary of these, skipping the fine details.

We define a *recursively presented transition system (r.p.t.s.)* to be a labelled transition system $(\mathscr{S}, \mathscr{L}, \longrightarrow)$ such that $\mathscr{S}$ is (recursively codable as) a recursive set of integers, $\mathscr{L}$ likewise, and $\longrightarrow$ is recursive. Henceforth we consider only recursively presented transition systems, with recursive valuations for the free variables. We have the following theorem, which is proved by a trivial translation of the semantics of the modal mu-calculus into mu-arithmetic:

Theorem 3 [Bra97] *For a modal mu-calculus formula $\Phi \in \Sigma_n^\mu$, the denotation $\|\Phi\|$ in any r.p.t.s. is a Σ_n^μ definable set of integers.* □

We also have the converse

Theorem 4 [Bra97] *Let $\phi(z)$ be a Σ^μ_n formula of mu-arithmetic. There is a r.p.t.s. $\mathscr{T}$ with recursive valuation $\mathscr{V}$ and a Σ^μ_n formula Φ of the modal mu-calculus such that $\phi((s)_0)$ iff $s \in \|\Phi\|^{\mathscr{T}}_{\mathscr{V}}$. (Thus if ϕ is not Σ^μ_{n-1}-definable, neither is $\|\Phi\|$.)* □

This theorem is not inherently difficult; it is established by coding the evaluation of mu-arithmetic formulae into a r.p.t.s. and a modal mu-calculus formula, in such a way that arithmetic computation is handled by the transitions of the system, and the fixpoints of ϕ are handled by the fixpoints of Φ. The proof is then a fairly straightforward induction. In this paper, we shall see a simplified version of this technique.

These two theorems establish the modal alternation hierarchy: use Theorem 4 to code an arithmetic strict Σ^μ_n set of integers by a strict Σ^μ_n modal mu-formula Φ on a r.p.t.s. $\mathscr{T}$; by Theorem 3, no Σ^μ_{n-1} modal formula can have the same denotation in $\mathscr{T}$, and so no Σ^μ_{n-1} modal formula is logically equivalent to Φ.

2.4 Tree algebras

Niwiński's papers [Niw86,Niw97] contain an extensive study of fixpoint algebras. For our purposes here, we consider just the most concrete versions, namely those over trees. Refer to [Niw97], which is an excellent exposition, for further details and for the generalizations which do not concern us here.

Let Σ be a *signature* containing a finite number of *operators* each with an arity. For example, take $\Sigma = \{a(-,-), b(-), c\}$, with one binary, one unary and one nullary operator. A *tree over* Σ is a possibly infinite tree with nodes labelled by operators, such that a node labelled by f has arity(f) children.

Define a fixpoint logic over Σ thus: variables Z, and tt and ff are formulae; conjunction $\wedge$ and disjunction $\vee$ of formulae are formulae; for each operator $f \in \Sigma$ with arity n and formulae $\phi_1, \ldots, \phi_n$, $f(\phi_1, \ldots, \phi_n)$ is a formula; for a formula ϕ with free variable Z, $\mu Z.\,\phi$ and $\nu Z.\,\phi$ are formulae.

Given a particular tree t, this logic is interpreted over the set of nodes of t in the obvious way: a node satisfies $f(\phi_1, \ldots \phi_n)$ if it is labelled by f and its children satisfy respectively $\phi_1, \ldots \phi_n$. The fixpoints are interpreted as

in the modal mu-calculus: a formula with free variable Z defines a function on the powerset lattice of nodes. We define the fixpoint alternation classes of formulae in the usual way.

We say that a tree satisfies ϕ if its root does. An important property of these logics is the 'internalization property' [Niw97]: given a tree t and a node n of t, the node n satisfies a formula ϕ iff the subtree of t rooted at n satisfies ϕ.

For examples of this logic with the signature above, we can consider the following. $\mu Z.\, a(Z,Z) \vee b(Z) \vee c$ defines the set of finite trees; $\nu Z.\, b(Z)$ defines the infinite linear tree $\bullet \xrightarrow{b} \bullet \xrightarrow{b} \cdots$; $\nu Y.\, \mu Z.\, a(c,Y) \vee a(Z,Z) \vee b(Z)$ defines the set of trees such that on every path there are infinitely many a nodes with a c left child, and c only occurs as the left child of a.

A tree algebra is then the set of all trees over a given signature, with the operations defined by the interpretations of the logical operators. We have no need to consider the algebraic view, and can stick to the logic. The problem left open by Niwiński can be stated as follows: is there a signature Σ such that the hierarchy of sets of trees definable by Σ_n^μ formulae is strict? For the case where intersection $\wedge$ is *not* a primitive symbol of the logic, [Niw86] showed strictness; however, the proof does not go through for the case with intersection, and indeed the exhibited hard Σ_n^μ formulae for the intersection-free case are in fact all equivalent to alternation depth 2 formulae with intersection.

3 A simple proof of the alternation hierarchy in mu-arithmetic

The first result of this paper is to observe that the alternation hierarchy theorem in mu-arithmetic can be proved simply along the lines of the proof of the strictness of the Kleene arithmetic hierarchy. The technique is to show that the truth of Σ_n^μ formulae can itself be expressed by a Σ_n^μ formula, and to use a diagonalization argument to show that this formula cannot be equivalent to any Π_n^μ formula.

Firstly, take a suitable Gödel numbering of mu-arithmetic. We consider only formulae without free set variables; wlog, we may assume that all encoded formulae are in normal form, and are normalized so that the free individual variables are $\mathsf{z}_0, \ldots, \mathsf{z}_k$, the first-order quantifiers bind $\mathsf{z}_{k+1}, \ldots$, and for a formula of alternation depth n, the fixpoint variables are $\mathsf{X}_n, \ldots, \mathsf{X}_1$, with associated individual variables $\mathsf{x}_n, \ldots, \mathsf{x}_1$. We use

sans-serif type to indicate that the variable is being seen as part of an encoded object-level formula; normal italic type indicates a meta-level variable. We use corner quotes to denote the Gödel coding. We also need coded *assignments* which map an encoded variable to a value: we write $[v/\mathsf{z}]$ for the assignment that maps z (strictly, the code $\ulcorner \mathsf{z} \urcorner$) to the integer v, and $a[v/\mathsf{z}]$ for the updating of a by $[v/\mathsf{z}]$. We use double quotes to indicate the appropriate meta-language formalization of the informal statement inside the quotes.

Now suppose that $\mathrm{Sat}_n(x, y)$ is a formula of mu-arithmetic expressing the truth of Σ^μ_n formulae, so that if ϕ is a Σ^μ_n formula and a an assignment of values $\vec{v}$ to the free variables $\vec{\mathsf{z}}$ of ϕ, then $\mathrm{Sat}_n(\ulcorner \phi \urcorner, a)$ is true just in case $\phi(\vec{v}/\vec{\mathsf{z}})$ is true. We have the

Lemma 5 $\mathrm{Sat}_n(z_0, [z_0/\mathsf{z}_0])$ *is not equivalent to any* Π^μ_n *formula.*

Proof. The proof is exactly as for the arithmetical hierarchy. Suppose the contrary, i.e. that $\neg\mathrm{Sat}_n(z_0, [z_0/\mathsf{z}_0])$ is equivalent to some Σ^μ_n formula $\theta(z_0)$. Then we have $\theta(\ulcorner \theta \urcorner)$ iff $\neg\mathrm{Sat}_n(\ulcorner \theta \urcorner, [\ulcorner \theta \urcorner/\mathsf{z}_0])$ iff $\neg\theta(\ulcorner \theta \urcorner)$, which is a contradiction. □

It remains to show that Sat_n exists and is indeed a Σ^μ_n formula.

Theorem 6 Sat_n *is a* Σ^μ_n *formula of mu-arithmetic, for* $n > 0$.

Proof. We start by constructing Sat_0, truth in first-order arithmetic, both as a Σ^μ_1 formula and as a Π^μ_1 formula. $\mathrm{Sat}_0(x, y)$ is defined as:

$$\begin{aligned}
\langle x, y\rangle \in \mu(w, W).\ &\text{“}(w)_0 = \ulcorner P(\tau) \urcorner \text{ and } \mathrm{pred}(\ulcorner P \urcorner, \mathrm{eval}(\ulcorner \tau \urcorner, (w)_1))\text{”} \\
&\vee \text{“}(w)_0 = \ulcorner \phi_1 \wedge \phi_2 \urcorner \text{ and } (\langle \ulcorner \phi_1 \urcorner, (w)_1\rangle \in W \wedge \langle \ulcorner \phi_2 \urcorner, (w)_1\rangle \in W)\text{”} \\
&\vee \text{“}(w)_0 = \ulcorner \phi_1 \vee \phi_2 \urcorner \text{ and } (\langle \ulcorner \phi_1 \urcorner, (w)_1\rangle \in W \vee \langle \ulcorner \phi_2 \urcorner, (w)_1\rangle \in W)\text{”} \\
&\vee \text{“}(w)_0 = \ulcorner \exists \mathsf{z}_i.\, \phi_1 \urcorner \text{ and } \exists v.\, \langle \ulcorner \phi_1 \urcorner, (w)_1[v/\mathsf{z}_i]\rangle \in W\text{”} \\
&\vee \text{“}(w)_0 = \ulcorner \forall \mathsf{z}_i.\, \phi_1 \urcorner \text{ and } \forall v.\, \langle \ulcorner \phi_1 \urcorner, (w)_1[v/\mathsf{z}_i]\rangle \in W\text{”}
\end{aligned}$$

where $\mathrm{eval}(t, a)$ is the recursive function which evaluates a coded term $t = \ulcorner \tau \urcorner$ under the variable assignment a, and $\mathrm{pred}(p, x)$ is the computable predicate which is true if the value x satisfies the predicate coded by $p = \ulcorner P \urcorner$.

We have here skipped the details of the coding, which are standard. For example, if we look in more detail at the clause for $\forall$, it actually says:

$$f((w)_0) = \ulcorner \forall \urcorner \wedge \forall v.\, \langle g((w)_0), h((w)_1, v, g'((w)_0)) \rangle \in W$$

where f extracts the top-level connective of a coded formula, g extracts the body of a $\forall$ formula and g' extracts the bound variable, and $h(a, v, z)$ takes the variable assignment a and updates the variable whose code is z by the value v. The fact that these functions f, g, h are recursive is obvious, and since we allow ourselves all recursive functions as primitives, that is sufficient; but explicit definitions in standard arithmetic may be found in references such as [Kay91].

It is clear that this fixpoint formula simply encodes directly the definition of truth in arithmetic. The formula is Σ_1^μ, but since the encoded recursive function terminates on all arguments—it is just a definition by induction on the structure of formulae—it does not matter whether we use a minimal or maximal fixpoint to achieve the recursion. Thus we may also obtain Sat_0 as a Π_1^μ formula.

In order to encode within mu-arithmetic the evaluation of formulae with fixpoints, it is necessary to have the same fixpoint structure in the Sat formula as in the formula it's evaluating. Recall that we assume pair-normal form, and suppose that we wish to evaluate Σ_n^μ formulae where n is odd, that is, formulae of the form

$$\tau_n \in \mu \mathsf{X}_n.\, \tau_{n-1} \in \nu \mathsf{X}_{n-1} \ldots . \tau_2 \in \nu \mathsf{X}_2.\, \tau_1 \in \mu \mathsf{X}_1.\, \phi \qquad (*)$$

where ϕ is first-order. The interpretation of the pure first-order part of ϕ may be done with the Σ_1^μ version of Sat_0—but ϕ may also now contain formulae $\tau \in \mathsf{X}_i$. We cannot code as integers the sets referred to by the X_i, so they must be represented by set variables in the meta-language. We use the meta-level variable X_i to represent the object variable X_i, and extend the body of Sat_0 by the clauses (for each $1 \leq i \geq n$)

$$\vee \text{ “}(w)_0 = \ulcorner \tau \in \mathsf{X}_i \urcorner \text{ and } \mathrm{eval}(\ulcorner \tau \urcorner, (w)_1)) \in X_i\text{”}.$$

Let Sat_0' denote the adjusted Sat_0.

With these adjustments, we have that $(*)$ is true with free variable assigment a just in case

$$\begin{aligned}
&\mathrm{eval}(\ulcorner\tau_n\urcorner, a) \in \mu X_n.\\
&\quad \mathrm{eval}(\ulcorner\tau_{n-1}\urcorner, a[x_n/\mathsf{x}_n]) \in \nu X_{n-1}.\ \ldots\\
&\qquad \mathrm{eval}(\ulcorner\tau_1\urcorner, a[x_n, \ldots, x_2/\mathsf{x}_n, \ldots, \mathsf{x}_2]) \in \mu X_1.\\
&\qquad\quad \mathrm{Sat}'_0(\ulcorner\phi\urcorner, a[x_n, \ldots, x_1/\mathsf{x}_n, \ldots, \mathsf{x}_1])
\end{aligned}$$

Now we just parametrize on $(*)$: let $f_1(x, y)$ be the function that given x encoding a Σ^μ_n formula $(*)$ and an assignment y, computes $\mathrm{eval}(\ulcorner\tau_n\urcorner, y)$, and so on, and let $g(x)$ extract the body of $(*)$. Then we have $\mathrm{Sat}_n(x, y)$ in the form

$$\begin{aligned}
&f_n(x, y) \in \mu X_n.\, f_{n-1}(x, y[x_n/\mathsf{x}_n]) \in \nu X_{n-1}.\ \ldots\\
&\quad f_1(x, y, [x_n, \ldots, x_2/\mathsf{x}_n, \ldots, \mathsf{x}_2]) \in \mu X_1.\\
&\qquad \mathrm{Sat}'_0(g(x), y[x_n, \ldots, x_1/\mathsf{x}_n, \ldots, \mathsf{x}_1])
\end{aligned}$$

which is Σ^μ_n as required. If n is even, we use the Π^μ_1 version of Sat_0 instead.

The fact that Sat_n does indeed code truth is easily shown: show by induction on i that each meta-level fixpoint set X_i coincides with the object-level set X_i. The base case follows from the correctness of Sat'_0, and the induction step is easy.

It may be noted that we have also skipped details of what the functions f_1 etc. should do if given ill-formed arguments. Any convenient trick may be used; the details are unimportant. □

4 The simple examples in the modal mu-calculus

To construct examples of strict Σ^μ_n formulae in the modal mu-calculus, it would suffice to apply the general construction of Theorem 4 to Sat_n. However, Sat_n contains a large number of function symbols, and the translation would contain many labels. By specializing the general construction, we can eliminate most of these labels, and obtain very simple examples. The following presentation is self-contained, but terse; for a longer explanation of the technique, see [Bra97].

We aim to construct a transition system $\mathscr{T}$ and a Σ_n^μ modal mu-calculus formula Φ such that the set of states satisfying Φ is defined by the strict Σ_n^μ arithmetic formula Sat_n.

The transition system $\mathscr{T}$ should be viewed as a machine for evaluating arithmetic expressions in the same way that Sat_n does: the computation happening in the body of Sat_0' will be dealt with by the definition of the transitions of $\mathscr{T}$, and the arithmetic fixpoints are translated into modal fixpoints in Φ.

The states of $\mathscr{T}$ encode several pieces of information. Namely, a state s contains: a formula ψ_s of the form $(*)$, and a variable assignment a_s, and a pointer p_s into ψ_s which keeps track of where we are in the evaluation. We use the notation of $(*)$ to refer to parts of ψ_s.

The labels of $\mathscr{T}$ are used to distinguish various steps of computation; we shall start with enough labels to make the construction clear, and then argue the number down a little. In the interests of clarity, we shall use roman letters for the modal labels: so x is a meta-level arithmetic variable, x is an object-level arithmetic variable, and x is a modal label associated with x.

The transitions of $\mathscr{T}$ from a state s are defined thus:

- If p_s points at τ_i (or after $\mu\mathsf{X}_{i+1}.$, which we consider to be the same), then $s \xrightarrow{\mathrm{x}_i} s'$ where $\psi_{s'} = \psi_s$, and $a_{s'} = a_s[\mathrm{eval}(\tau_i, a_s)/\mathsf{x}_i]$ and $p_{s'}$ points after $\mu\mathsf{X}_i$. That is, the term τ_i is evaluated in the current assignment, x_i is set to its value, and we start evaluating the inner fixpoint.

Otherwise, p_s points at a subformula of ϕ. The transition from s mimics the appropriate clause of Sat_0'. The ψ component is not altered by any transition, and the a component is unchanged unless otherwise stated.

- If p_s points at $P(\tau)$, then $s \xrightarrow{\mathrm{a}} s_\mathrm{a}$ ('a' for atom), where s_a is a special state with no structure, only if $P(\tau)$ is true with variable assignment a_s; otherwise there are no transitions from s.
- If p_s points at $\phi_1 \wedge \phi_2$, then $s \xrightarrow{\mathrm{c}} s_k$ ('c' for conjunction) for $k = 1, 2$, where p_{s_k} points at ϕ_k.
- If p_s points at $\forall \mathsf{z}_i.\, \phi_1$, then $s \xrightarrow{\mathrm{c}} s_k$ (universal quantification is treated as conjunction) for $k \in \mathbb{N}$, where p_{s_k} points at ϕ_1, and $a_{s_k} = a_s[k/\mathsf{z}_i]$.
- If p_s points at $\phi_1 \vee \phi_2$, then $s \xrightarrow{\mathrm{d}} s_k$ ('d' for disjunction) for $k = 1, 2$, where p_{s_k} points at ϕ_k.
- If p_s points at $\exists \mathsf{z}_i.\, \phi_1$, then $s \xrightarrow{\mathrm{d}} s_k$ (existential quantification is treated as disjunction) for $k \in \mathbb{N}$, where p_{s_k} points at ϕ_1, and $a_{s_k} = a_s[k/\mathsf{z}_i]$.

- If p_s points at $\tau \in \mathsf{X}_i$, then $s \xrightarrow{\mathrm{x}_i} s'$, where $p_{s'}$ points after $\mu\mathsf{X}_i.$, and $a_{s'} = a_s[\mathrm{eval}(\tau, a_s)/\mathsf{x}_i]$. That is, τ is evaluated, copied to the input variable x_i of the fixpoint X_i, and evaluation of the fixpoint started.

It is clear that $\mathscr{T}$ is a recursively presented transition system.

Now consider the following *modal* mu-calculus formula:

$$\begin{aligned}\mathrm{MuSat}_n \stackrel{\mathrm{def}}{=}\ & \langle \mathrm{x}_n\rangle \mu X_n.\, \langle \mathrm{x}_{n-1}\rangle \nu X_{n-1}.\, \ldots \langle \mathrm{x}_1\rangle \mu X_1.\, \mu W. \\ & \langle \mathrm{a}\rangle \mathrm{tt} \ \vee\ (\langle \mathrm{c}\rangle \mathrm{tt} \wedge [\mathrm{c}]W) \ \vee\ \langle \mathrm{d}\rangle W \\ & \vee\ \langle \mathrm{x}_1\rangle X_1 \ \vee\ \ldots \ \vee\ \langle \mathrm{x}_n\rangle X_n\end{aligned}$$

By the construction of $\mathscr{T}$, we have:

Theorem 7 *$s \models \mathrm{MuSat}_n$ just in case p_s points at ψ_s, and $\mathrm{Sat}_n(\ulcorner\psi_s\urcorner, a_s)$. Hence MuSat_n is a strict Σ^μ_n modal formula.*

Proof. The proof is a special case of the proof of Theorem 4, the details of which are in [Bra97]. □

MuSat_n is already quite a simple formula, but it is interesting to try to simplify it further, which we shall do in stages.

Firstly, is it necessary to have the double occurrence of $\langle \mathrm{x}_i\rangle$, or can we remove the guards from the fixpoint formulae? Yes, we can: consider the formula

$$\mathrm{MuSat}'_n \stackrel{\mathrm{def}}{=} \mu X'_n.\, \nu X'_{n-1}.\, \ldots \mu X'_1.\, \mu W.\, \Psi$$

where Ψ is formed from the body of MuSat_n by priming the X_is. The relation between MuSat_n and MuSat'_n is that $X'_n = \ldots = X'_1 = X_1$) (note that in MuSat_n, we have $X_1 \supseteq X_2 \cup \ldots \cup X_n$), and conversely $X_i = \langle \mathrm{x}_i\rangle X'_i$ for $i = n, \ldots, 2$. The denotation of MuSat'_n is still a strict Σ^μ_n set, since the denotation of MuSat_n is a projection of it.

Next, the occurrence of $\langle \mathrm{c}\rangle \mathrm{tt}$ is irritating. Its purpose is to assert that p_s is indeed pointing at an $\wedge$-subterm of ψ_s—of course, $[\mathrm{c}]W$ is true at any state with no c-transitions from it. However, we can render it unnecessary by modifying $\mathscr{T}$: if s is any state *other than* an $\wedge$-subterm state, then add a transition $s \xrightarrow{\mathrm{c}} s$. Since W is a least fixpoint variable, if W is true at a state with a c-loop, it is true by virtue of some other disjunct than $[\mathrm{c}]W$, and it is not true if it was not true before the loop was added.

We can also eliminate the requirement for a separate a-transition, by modifying the modification: for all those states s with an a-transition, remove the c-loop added in the previous paragraph; now $[\mathrm{c}]W$ is true at those states, so we can discard the $\langle \mathrm{a} \rangle \mathrm{tt}$ clause.

Finally, we note that $W = X'_1$, and they are adjacent least fixpoints, so we can amalgamate them; further, the job of the d transition can as well be done by x_1, since they work on disjoint sets of states.

Hence we arrive at the following very simple example of a strict Σ^μ_n modal formula (replacing X' by X again):

$$\mathrm{MuSat}''_n \stackrel{\mathrm{def}}{=} \mu X_n . \nu X_{n-1} . \ldots \mu X_1 . [\mathrm{c}] X_1 \vee \langle \mathrm{x}_1 \rangle X_1 \vee \ldots \vee \langle \mathrm{x}_n \rangle X_n$$

5 Relation to parity games.

In the earliest version of this paper, we showed that if one chooses to look at models with no action labels, but with atomic propositions, the above formula appears in a form that is the same as the formula describing the existence of a winning strategy in a parity game of rank n, and hence that formula is strict Σ^μ_n.

When this paper was submitted to STACS, one of the referees pointed out that the strictness of the winning strategy formula can be shown directly from [Bra97] and the game interpretation of modal mu-calculus [EmJ91], without requiring the explicit use of the arithmetic formula Sat_n. As this is an elegant proof, we outline it here, and then comment on the similarities to MuSat.

A parity game of rank n [EmJ91] is played on a directed graph with the following properties: every vertex belongs either to Player or Opponent, and every vertex has an index between 1 and n. If the current vertex belongs to player A, then A moves by choosing a successor vertex. (In [EmJ91], the graph is bipartite so that Player and Opponent alternate, but this is not essential.) In a given play, Player wins if either Opponent gets stuck, or if the greatest index occurring infinitely often is even. ('greatest' is sometimes replaced by 'least', e.g. in [Wal96].) For simplicity, assume henceforth that n is odd.

Given such a graph, let P be true at Player vertices, O be true at Opponent vertices, and R_i true at vertices of index i. It is easy to show

[EmJ91,Wal96] that the modal mu-calculus formula

$$\mathrm{Par}_n \stackrel{\mathrm{def}}{=} \mu X_n.\, \nu X_{n-1}.\, \ldots \mu X_1.$$
$$\big(P \Rightarrow \langle\rangle \textstyle\bigwedge_{1\le i\le n}(R_i \Rightarrow X_i)\big) \wedge \big(O \Rightarrow [] \textstyle\bigwedge_{1\le i\le n}(R_i \Rightarrow X_i)\big)$$

defines exactly the set of vertices from which Player has a winning strategy.

Now take a strict Σ^μ_n formula $\phi(z)$ of mu-arithmetic, and construct the r.p.t.s. $\mathscr{T}$ and modal formula Φ of Theorem 4. Given a Σ^μ_n modal formula, one can easily, and recursively, construct a parity game G of rank n, whose vertices are pairs (s, Ψ) of states of $\mathscr{T}$ and subformulae of Φ, such that Player wins from (s, Ψ) iff $s \models^{\mathscr{T}} \Psi$. Hence $(s, \Phi) \models^G \mathrm{Par}_n$ iff $\phi(s_0)$. Therefore $\|\mathrm{Par}_n\|^G$ is an arithmetic Σ^μ_n-hard set, and so by Theorem 3 we conclude

Theorem 8 Par_n *is a strict* Σ^μ_n *modal formula.* □

The alternative approach for showing the strictness of Par_n is to work from the transition system $\mathscr{T}$ of Theorem 7, and replace the action labels by atomic propositions, so that P is true at disjunctive states, O at conjunctive states, and R_i at X_i states. With a little manipulation along the lines of the construction of MuSat''_n from MuSat_n, one obtains exactly the formula Par_n as the modal encoding of Sat_n. Thus we use Sat_n explicitly, and use a specialization of the proof of Theorem 4. The STACS referee's suggestion avoids this work, and so produces the Par_n examples from [Bra97] without using the simple proof of the mu-arithmetic hierarchy.

6 Modal mu-calculus alternation on bounded-branching systems.

Our next step is to show that the alternation hierarchy remains strict even on bounded-branching systems, and in particular on systems with a maximum branching degree of 2.

Consider the formula

$$\mathrm{Hard}_n \stackrel{\mathrm{def}}{=} \mu X_n.\, \nu X_{n-1}.\, \ldots \mu\nu X_1.\ \ \langle \mathrm{a}\rangle \mathrm{tt} \vee (\langle \mathrm{c}\rangle \mathrm{tt} \wedge [\mathrm{c}] X_1)$$
$$\vee \langle \mathrm{d}\rangle X_1 \vee \langle \mathrm{x}_1\rangle X_1 \vee \ldots \vee \langle \mathrm{x}_n\rangle X_n$$

which is the same as MuSat_n except that we have amalgamated the adjacent fixpoints X_1 and W. The transition system on which we considered

this formula had the convenient property that every state has outgoing transitions of only one label. However, many states had infinitely many successors; we need to address this. To solve Niwiński's problem, we shall also need to deal with the issue that the formula uses $n+2$ different labels; but this is a simple encoding issue.

For reasons that will become apparent when we consider Niwiński's trees, we should like to have a transition system with exactly two labels, l, r such that every state has no successor or one l and one r successor.

Using standard techniques, we build a new transition system $\mathscr{T}'_n$ in which the new labels code the old labels. For concreteness, let us say that if $s \xrightarrow{\alpha} t$ in $\mathscr{T}_n$, where α is a or x_i, then becomes

$$\begin{array}{ll} s & \xrightarrow{l} t \\ \downarrow r & \\ T_\alpha & \end{array}$$

and if $s \xrightarrow{\beta} t_i$, where β is c or d, and $1 \leq i \leq k$ for $k = 2$ or $k = \infty$, according as s has two successors (when coding a conjunction or disjunction) or infinitely many successors (when coding a box or diamond), then

$$\begin{array}{lllllll} s & \xrightarrow{l} & u_1 & \xrightarrow{l} & \cdots & \xrightarrow{l} & u_k \xrightarrow{l} u_0 \\ \downarrow r & & \downarrow r & & \cdots & & \downarrow r \\ T_\beta & & t_1 & & \cdots & & t_k \end{array}$$

where the u states are new 'junk' states, and T_α is a particular finite binary tree coding the label. For example, T_a might be the binary tree $\bullet$, T_c the tree $(\bullet, \bullet)$, T_d the tree $((\bullet, \bullet), \bullet)$, and so on. Let Ψ_α be a modal formula characterizing T_α: for example, Ψ_c would be $\langle l\rangle[]\mathrm{ff} \vee \langle r\rangle[]\mathrm{ff}$.

We now need to translate the formula Hard_n for the new system. Obviously we can translate $\langle \mathrm{x}_i\rangle\Phi$ into $\langle r\rangle\Psi_{\mathrm{x}_i} \wedge \langle l\rangle\Phi$ etc., but the translation of the c and d modalities requires introducing additional fixpoints. The (sometimes infinite) branching box $\langle \mathrm{c}\rangle\mathrm{tt} \wedge [\mathrm{c}]X_1$ becomes $\langle r\rangle\Psi_\mathrm{c} \wedge [l]\nu Y.\, [l]Y \wedge [r]X_1$; and the branching diamond $\langle \mathrm{d}\rangle X_1$ becomes $\langle r\rangle\Psi_\mathrm{d} \wedge \langle l\rangle\mu Y.\, \langle r\rangle X_1 \vee \langle l\rangle Y$.

Since the construction of $\mathscr{T}'_n$ is recursive, the new formula Hard'_n still denotes an arithmetic Σ^μ_n-hard set of states. Unfortunately, Hard'_n is now a modal Σ^μ_{n+1} formula, owing to the introduction of the new fixpoints! This is still sufficient to establish the hierarchy, since it cannot be equivalent to

any modal Π^μ_n formula. However, we could also obtain a direct proof that it cannot be equivalent to any modal Π^μ_{n+1} formula, since it can be shown, by extending an analysis of [Bra96], that on a bounded degree recursive transition system a modal Σ^μ_n formula denotes a set of at most arithmetic Σ^μ_{n-1} complexity.

We now have the result

Theorem 9 *There is a class of transition systems with branching degree* ≤ 2 *on which the modal mu-calculus alternation hierarchy is strict.*

7 Alternation in tree algebras.

Owing to the way in which we have set up $\mathscr{T}'_n$, the transfer to tree algebras is almost immediate. We take the signature Σ with a binary function symbol a and a nullary function symbol c. The left and right children of a node labelled a correspond to the l and r successors of a non-leaf node in $\mathscr{T}_n$; thus a tree over this signature is a transition system of the form specified for $\mathscr{T}'_n$, and conversely the unwinding $\mathscr{T}''_n$ of $\mathscr{T}'_n$ is a tree over this signature. Since unwinding is a recursive operation, we can say that given a Σ^μ_n tree mu-formula, the set of nodes of $\mathscr{T}''_n$ satisfying it is of arithmetic complexity Σ^μ_{n-1}. However, we can translate Hard'_n into a tree formula Hard''_n:

$$\begin{aligned}\mu X_n.\ \ldots \mu\nu X_1.\quad & a(\mathrm{tt}, \psi_{\mathrm{a}}) \\ & \vee\, a(\nu Y.\, a(Y, X_1), \psi_{\mathrm{c}}) \\ & \vee\, a(\mu Y.\, a(Y, \mathrm{tt}) \vee a(\mathrm{tt}, X_1), \psi_{\mathrm{d}}) \\ & \vee\, a(X_1, \psi_{\mathrm{x}_1}) \vee \ldots \vee a(X_n, \psi_{\mathrm{x}_n})\end{aligned}$$

where ψ_α is the translation of Ψ_α, for example $\psi_{\mathrm{a}} = c, \psi_{\mathrm{c}} = a(c, c), \psi_{\mathrm{d}} = a(a(c, c), c)$ and so on. Hence the nodes of $\mathscr{T}''_n$ satisfying Hard''_n are exactly those that are the unwindings of states of $\mathscr{T}'_n$ satisfying Hard'_n; and hence this set is arithmetic Σ^μ_n-hard. It follows that for any formula ϕ of lower alternation depth, there is a node on which ϕ and Hard''_n disagree. Now by the crucial fact that a node satisfies a formula iff the subtree rooted at

that node satisfies it (the 'internalization property' of [Niw97]), there is a tree on which ϕ and Hard''_n disagree. Hence we have

Theorem 10 *The fixpoint alternation hierarchy over the Niwiński tree logic with signature* $\{a(-,-), c\}$ *and intersection is strict.*

Of course, as for the intersection-free result of [Niw97], this signature is the minimal signature: we need at least one constant with arity 2, and at least one other symbol. With a more generous signature, the coding is less messy, and simpler hard formulae can be presented, as was done in [Bra98a] (for the signature $\{a(-,-), b(-)\}$).

It is interesting to note that even though we are establishing a hierarchy with intersection, the hard formulae do not themselves use intersection once they are expressed in the tree logic; they need only the 'implicit' intersection given by the binary symbol a. However, unlike the hard formulae of [Niw97], which are in fact all equivalent to level Σ_2^μ formulae in the intersection-full hierarchy, the implicit conjunction conjoins different fixpoints. One may also compare the disjunctive formulae of [JaW95].

8 Remarks

Although this approach has solved the problem of Niwiński, one might reasonably object to it on aesthetic grounds. It should be possible to solve a problem about a fairly weak logic on the binary tree without resorting to the use of arithmetic and Gödel encodings. Indeed, one might hope that the diagonalization argument, used to prove this hierarchy as many others, could be carried out directly on the trees. This hope is not vain: by the time this work was presented at FICS, André Arnold [Arn9?] had discovered an elegant technique which uses a form of diagonalization on the binary tree, together with some basic (ultra-)metric space theory, to show the hardness of all the example formulae produced by myself, Lenzi, and others.

9 Acknowledgements

I especially thank Alex Simpson, who suggested to me that there must be a simple proof of the mu-arithmetic hierarchy along these lines. Thanks also to Igor Walukiewicz, who pointed me at the parity game formulae. In

addition to the referee who provided the improved proof for parity game formulae, other STACS referees provided helpful suggestions; I am grateful to them. André Arnold pointed out a minor lacuna in [Bra98a].

I am supported by an Advanced Fellowship from the United Kingdom Engineering and Physical Sciences Research Council; also BRICS, the Danish National Research Foundation Centre for Basic Research In Computer Science, is supporting my visit to Aarhus.

Version Control: rairo.tex:1.4; suppprelims.tex:1.8; niwprelims.tex:1.2; suppmain.tex:1.9; suppparity.tex:1.2.

References

[Arn9?] A. Arnold, *paper submitted for publication*

[Bra91] J. C. Bradfield, *Verifying Temporal Properties of Systems* (Birkhäuser, Boston, 1991).

[Bra96] J. C. Bradfield, On the expressivity of the modal mu-calculus, in: C. Puech and Rüdiger Reischuk, eds., *Proc. STACS '96*, LNCS **1046** (Springer, Berlin, 1996) 479–490.

[Bra97] J. C. Bradfield, The modal mu-calculus alternation hierarchy is strict, *Theor. Comput. Sci.* **195** 133–153 (1997).

[Bra98] J. C. Bradfield, Simplifying the modal mu-calculus alternation hierarchy, in: M. Morvan, C. Meinel and D. Krob, eds., *Proc. STACS 98*, LNCS **1373** (Springer, Berlin, 1998) 39–49.

[Bra98a] J. C. Bradfield, Fixpoint alternation on the binary tree, *Workshop on Fixpoints in Computer Science (FICS)*, Brno, 1998.

[EmJ91] E. A. Emerson and C. S. Jutla, Tree automata, mu-calculus and determinacy, in: *Proc. FOCS 91.* (1991)

[EmL86] E. A. Emerson and C.-L. Lei, Efficient model checking in fragments of the propositional mu-calculus, in: *Proc. 1st LICS* (IEEE, Los Alamitos, CA, 1986) 267–278.

[JaW95] D. Janin and I. Walukiewicz, Automata for the μ-calculus and related results, in *Proc. MFCS '95*, LNCS **969** (Springer, Berlin, 1995) 552–562.

[Kay91] R. Kaye, *Models of Peano Arithmetic.* (Oxford University Press, Oxford, 1991).

[Koz83] D. Kozen, Results on the propositional mu-calculus, *Theoret. Comput. Sci.* **27** (1983) 333–354.

[Len96] G. Lenzi, A hierarchy theorem for the mu-calculus, in: F. Meyer auf der Heide and B. Monien, eds., *Proc. ICALP '96*, LNCS **1099** (Springer, Berlin, 1996) 87–109.

[Lub93] R. S. Lubarsky, μ-definable sets of integers, *J. Symbolic Logic* **58** (1993) 291–313.

[Niw86] D. Niwiński, On fixed point clones, in: L. Kott, ed., *Proc. 13th ICALP*, LNCS **226** (Springer, Berlin, 1986) 464–473.

[Niw97] D. Niwiński, Fixed point characterization of infinite behavior of finite state systems. *Theoret. Comput. Sci.* **189** (1997) 1–69.

[Sti91] C. P. Stirling, Modal and temporal logics, in: S. Abramsky, D. Gabbay and T. Maibaum, eds., *Handbook of Logic in Computer Science*, Vol. 2 (Oxford University Press, 1991) 477–563.

[Wal96] I. Walukiewicz, Monadic second order logic on tree-like structures, in: C. Puech and Rüdiger Reischuk, eds., *Proc. STACS '96*, LNCS **1046** (Springer, Berlin, 1996) 401–414.

Recent BRICS Report Series Publications

RS-98-53 **Julian C. Bradfield.** ***Fixpoint Alternation: Arithmetic, Transition Systems, and the Binary Tree.*** **December 1998. 20 pp.**

RS-98-52 **Josva Kleist and Davide Sangiorgi.** ***Imperative Objects and Mobile Processes*****. December 1998. 22 pp. Appears in Gries and de Roever, editors,** ***IFIP Working Conference on Programming Concepts and Methods*****, PROCOMET '98 Proceedings, 1998, pages 285–303.**

RS-98-51 **Peter Krogsgaard Jensen.** ***Automated Modeling of Real-Time Implementation*****. December 1998. 9 pp. Appears in** ***The 13th IEEE Conference on Automated Software Engineering*****, ASE '98 Doctoral Symposium Proceedings, 1998, pages 17–20.**

RS-98-50 **Luca Aceto and Anna Ingólfsdóttir.** ***Testing Hennessy-Milner Logic with Recursion*****. December 1998. 15 pp. Appears in Thomas, editor,** ***Foundations of Software Science and Computation Structures: Second International Conference*****, FoSSaCS '99 Proceedings, LNCS 1578, 1999, pages 41–55.**

RS-98-49 **Luca Aceto, Willem Jan Fokkink, and Anna Ingólfsdóttir.** ***A Cook's Tour of Equational Axiomatizations for Prefix Iteration*****. December 1998. 14 pp. Appears in Nivat, editor,** ***Foundations of Software Science and Computation Structures: First International Conference*****, FoSSaCS '98 Proceedings, LNCS 1378, 1998, pages 20–34.**

RS-98-48 **Luca Aceto, Patricia Bouyer, Augusto Burgueño, and Kim G. Larsen.** ***The Power of Reachability Testing for Timed Automata*****. December 1998. 12 pp. Appears in Arvind and Ramanujam, editors,** ***Foundations of Software Technology and Theoretical Computer Science: 18th Conference*****, FST&TCS '98 Proceedings, LNCS 1530, 1998, pages 245–256.**

RS-98-47 **Gerd Behrmann, Kim G. Larsen, Justin Pearson, Carsten Weise, and Wang Yi.** ***Efficient Timed Reachability Analysis using Clock Difference Diagrams*****. December 1998. 13 pp. To appear in** ***Computer-Aided Verification: 11th International Conference*****, CAV '99 Proceedings, LNCS, 1999.**