Modal Mu-Calculi

Julian Bradfield and Colin Stirling

Contents

1 Introduction

Modal mu-calculus is a logic used extensively in certain areas of computer science, but also of considerable intrinsic mathematical and logical interest. Its defining feature is the addition of inductive definitions to modal logic; thereby it achieves a great increase in expressive power, and an equally great increase in difficulty of understanding. It includes many of the logics used in systems verification, and is quite straightforward to evaluate. It also provides one of the strongest examples of the connections between modal and temporal logics, automata theory and the theory of games.

In this chapter, we survey a range of the questions and results about the modal mu-calculus and related logics. For the most part, we remain at survey level, giving only outlines of proofs; but in places, determined partly by our own interests and partly by our sense of which problems have been – or had been – the longest-standing thorns in the side of the mu-calculus community, we go into more detail.

We start with an account of the historical context leading to the introduction of the modal mu-calculus. Then we define the logic formally, describe some approaches to gaining an intuitive understanding of formulae, and establish the main theorem about the semantics. Following that, we discuss how the modal mu-calculus has the tree model property and relates to some other temporal logics, to automata and to games. Next, an account of decidability is given – this is one of the thorns, at least for those who find automata prickly. We then consider briefly completeness, bisimulation invariance and the concept of fixpoint alternation, which plays a part in several interesting questions about the logic. Finally, we look at some generalizations of the logic.

Notation:
$L\mu$ means the modal mu-calculus, considered as a logical language (not as a theory). In general, the notation follows as much as possible the standards for this book, but because $L\mu$ is mostly studied in a setting with rather different traditions, and because we also need to notate several other concepts, we have made some compromises. Few of these should cause any difficulty, but let us note the following. Since $\rightarrow$ is often used to represent the transition relation in models (alias the accessibility relation from modal logic), we use $\Rightarrow$ rather than $\rightarrow$ for boolean implication. Structures, frames and models for $L\mu$ are usually viewed as transition systems, and so are usually called $\mathfrak{T}$ with state space $\mathfrak{S}$. States within systems (i.e. worlds in the language of modal logic) are typically s, t, whereas p, q, r are states in an automaton. Hence we write atomic propositions with capital $P, Q, \ldots$ rather than $p, q, \ldots$, and similarly variables ranging over sets of states are written X, Y.

2 Contextual background

The modal mu-calculus comes not from the philosophical tradition of modal logic, but from the application of modal and temporal logics to program verification. In this section, we outline the historical context for $L\mu$.

2.1 Modal logics in program verification

The application of modal and temporal logics to programs is part of a line of program verification going back to the 1960s and program schemes and Floyd–Hoare logic. Originally the emphasis was on *proof*: Floyd–Hoare logic allows one to make assertions about programs, and there is a proof system to verify these assertions. This line of work has, of course, continued and flourished, and today there are highly sophisticated theories for proving properties of programs, with equally sophisticated machine support for these theories. However, the use of proof systems has some disadvantages, and one hankers after a more purely algorithmic approach to simple problems. One technique was pioneered by Manna and Pnueli [48], who turned program properties into questions of satisfiability or validity in first order logic, which can then be attacked by means that are not just proof-theoretic; this idea was later applied by them to linear temporal logics.

During the 1970s, the theory of program correctness was extended by investigating more powerful logics, and studying them in a manner more similar to the traditions of mathematical logic. A family of logics which received much attention was that of dynamic logics, which can be seen as extending the ideas of Hoare logic [57]. Dynamic logics are modal logics, where the different modalities correspond to the execution of different programs—the formula $\langle\alpha\rangle\phi$ is read as 'it is possible for α to execute and result in a state satisfying ϕ'. The programs may be of any type of interest; the variety of dynamic logic most often referred to is a propositional language in which the programs are built from atomic programs by regular expression constructors; henceforth, Propositional Dynamic Logic, PDL, refers to this logic. PDL is interpreted with respect to a model on a Kripke structure, formalizing the notion of the global state in which programs execute and which they change—each point in the structure corresponds to a possible state, and programs determine a relation between states giving the changes effected by the programs.

Once one has the idea of a modal logic defined on a Kripke structure, it becomes quite natural to think of the finite case and write programs which just check whether a formula is satisfied. This idea was developed in the early 80s by Clarke, Emerson, Sistla and others. They worked with a logic that has much simpler modalities than PDL—in fact, it has just a single 'next state' modality—but which has built-in temporal connectives such as 'until'. This logic is CTL, and it and its extensions remain some of the most popular logics for expressing properties of systems.

Meanwhile, the theory of process calculi was being developed in the late 70s, most notably by Milner [50]. An essential component was the use of labelled Kripke structures ('labelled transition systems') as a raw model of concurrent behaviour. An important difference between the use of Kripke structures here and their use in program correctness was that the states are the behaviour expressions themselves, which model concurrent systems, and the labels on the accessibility relation (the transitions) are simple actions (and not programs). The criterion for behavioural equivalence of process expressions was defined in terms of observational equivalence (and later in terms of bisimulation relations). Hennessy and Milner introduced a primitive modal logic in which the modalities refer to actions: $\langle a\rangle\phi$ 'it is possible to do an a action and then have ϕ be true', and its dual $[a]\phi$ 'ϕ holds after every a action'. Together with the usual boolean connectives, this gives Hennessy–Milner logic [31], HML, which was introduced as an alternative exposition of observational equivalence. However, as a logic HML is obviously inadequate to express many properties, as it has no means of saying 'always in the future' or other temporal connectives—except by allowing infinitary conjunction. Using an infinitary logic is undesirable both for the obvious reason that infinite formulae are not amenable to automatic processing, and because infinitary logic gives much more expressive power than is needed to express temporal properties.

In 1983, Dexter Kozen published a study of a logic that combined simple modalities, as in HML, with fixpoint operators to provide a form of recursion. This logic, the modal mu-calculus, has become probably the most studied of all temporal logics of programs. It has a simple syntax, an easily given semantics, and yet the fixpoint operators provide immense power. Most other temporal logics, such as the CTL family, can be seen as fragments of $L\mu$. Moreover, this logic lends itself to transparent model-checking algorithms.

Another 'root' to understanding modal logics is the work in the 60s on automata over infinite words and trees by Büchi [13] and Rabin [60]. The motivation was decision questions of monadic second-order logics. Büchi introduced automata as a normal form for such formulae. This work founded new connections to logic and automata theory. Later it was realised that modal logics are merely sublogics of appropriate monadic second-order logic, and that the automata normal forms provide a very powerful framework within which to study properties of modal logic. A further development was the use of games by Gurevich and Harrington [30] as an alternative to automata.

There is also an older game-theoretic tradition due to Ehrenfeucht and Fraïssé, for understanding the expressive power of logics. These techniques are also applicable within process calculi. For instance bisimulation equivalence can be naturally rendered as such a game, and expressivity of modal logics can be understood using game-theoretic techniques.

2.2 Precursors to modal mu-calculus

HML, Hennessy–Milner Logic [31], is a primitive modal logic of action. The syntax of HML has, in addition to the boolean operators, a modality $\langle a\rangle\phi$, where a is a process *action*. A structure for the logic is just a labelled transition system. Atomic formulas of the logic are the constants $\top$ and $\bot$. The meaning of $\langle a\rangle\phi$ is 'it is possible to do an a-action to a state where ϕ holds'. The formal semantics is given in the obvious way by inductively defining when a state of a transition system, or a state of a process, has a property; for example, $s \models \langle a\rangle\phi$ iff $\exists t.s \xrightarrow{a} t \wedge t \models \phi$. We may also add some notion of variable or atomic proposition to the logic, in which case we provide a valuation which maps a variable to the set of states at which it holds. The expressive power of HML in this form is quite weak: obviously a given HML formula can only make statements about a given finite number of steps into the future. HML was introduced not so much as a language to express properties, but rather as an aid to understanding process equivalence: two (image-finite) processes are equivalent iff they satisfy exactly the same HML formulae. To obtain the expressivity desired in practice, we need stronger logics.

The logic PDL, Propositional Dynamic Logic [57,25], as mentioned above, is both a development of Floyd–Hoare style logics, and a development of modal logics. Recently, it has been used as a basis for description logics and logics of information. PDL is an extension of HML in the circumstance that the action set has some structure. There is room for variation in the meaning of action, but in the standard logic, a program is considered to have a number of *atomic actions*, which in process algebraic terms are just process actions, and α is allowed to be a regular expression over the atomic actions: a, $\alpha;\beta$, $\alpha \cup \beta$, or $\alpha*$. We may consider atomic actions to be uninterpreted atoms; but in the development from Floyd–Hoare logics, one would see the atomic actions as, for example, assignment statements in a **while** program.

PDL enriches the labels in the modalities of HML. An alternative extension of HML is to include further modalities. The branching time logic CTL, Computation Tree Logic [14], can be described in this way as an extension of HML, with some extra 'temporal' operators which permit expression of liveness and safety properties. For the semantics we need to consider 'runs' of a system. A run from an initial state or process s_0 is a sequence $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots$ which may have finite or infinite length; if it has finite length then its final process is a 'sink' process which has no transitions. A run $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots$ has the property $\phi\,\mathbf{U}\,\psi$, 'ϕ until ψ', if there is an $i \geq 0$ such that $s_i \models \psi$ and for all $j : 0 \leq j < i$, $s_j \models \phi$.

$$\begin{array}{llll}
s_0 \xrightarrow{a_1} & s_1 \xrightarrow{a_2} & \ldots\; s_i & \xrightarrow{a_{i+1}} \ldots \\
\models & \models & \ldots \models & \\
\phi & \phi & \quad\;\psi &
\end{array}$$

The formula $\mathbf{F}\phi = (\top\;\mathbf{U}\;\phi)$ means 'ϕ eventually holds' and $\mathbf{G}\phi = \neg(\top\;\mathbf{U}\;\neg\phi)$: '$\phi$ always holds'. For each 'temporal' operator such as $\mathbf{U}$ there are two modal variants, a strong variant ranging over all runs of a process and a weak variant ranging over some run of a process. We preface a strong version with $\forall$ and a weak version with $\exists$. If HML is extended with the two kinds of until operator the resulting logic is a slight but inessential variant of CTL (CTL does not in its standard form mention action labels in modalities). The formal semantics is given by inductively defining when a state (process) has a property. For instance $s \models \forall[\phi\,\mathbf{U}\,\psi]$ iff every run of s has the property $\phi\,\mathbf{U}\,\psi$.

CTL has variants and enrichments such as CTL* [24] and ECTL* [70]. These allow free mixing of path operators and quantifiers: for example, the CTL* formula $\forall[P\,\mathbf{U}\,\exists\mathbf{F}Q]$ is also a CTL formula, but $\forall[P\,\mathbf{U}\,\mathbf{F}Q]$ is not, because the $\mathbf{F}$ is not immediately governed by a quantifier. Hence extensions also cover traditional temporal logics—that is, literally logics of time—as advocated by Manna and Pnueli and others. In this view, time is a linear sequence of instants, corresponding to the states of just one execution path through the program. One can define a logic on paths which has operators $\bigcirc\phi$ 'in the next instant (on this path) ϕ is true', and $\phi\,\mathbf{U}\,\psi$ 'ϕ holds until ψ holds (on this path)'; and then a system satisfies a formula if all execution paths satisfy the formula—in CTL* terms, the specification is a path formula with a single outermost universal quantifier. One can also extend PDL with temporal operators, as in process logic.

There are extensions of all these logics to cover issues such as time and probability. The introduction of such real-valued quantities poses a number of problems, and such logics are still under active development.

2.3 The small model property

A general result about many modal logics is that they have the small model property; that is if a formula is satisfiable, then it is satisfiable by a model of some bounded size. Provided that model-checking is decidable, which is the case for almost all temporal logics, this immediately gives decidability of satisfiability for the logic, as one need simply check every transition system up to the size bound.

The technique used to establish the small model property for PDL (and therefore HML) is a classical technique in modal logic, that of *filtration*. Let s be a state, satisfying property ϕ, in a possibly infinite transition system $\mathfrak{T}$. Let Γ be the set of all subformulas of ϕ and their negations: in the case of PDL one also counts $\langle\alpha\rangle\psi$ and $\langle\beta\rangle\psi$ as subformulas of $\langle\alpha \cup \beta\rangle\psi$, $\langle\beta\rangle\psi$ as a subformula of $\langle\alpha;\beta\rangle\psi$ and $\langle\alpha;\alpha^*\rangle\psi$, and $\langle\alpha\rangle\psi$ as subformulas of $\langle\alpha^*\rangle\psi$. The size of Γ is proportional to $|\phi|$ (the length of ϕ). One then filters $\mathfrak{T}$ through Γ by defining an equivalence relation on the states of $\mathfrak{T}$, $s \equiv t$ iff $\forall\psi \in \Gamma . s \models \psi \Leftrightarrow t \models \psi$. We define the filtered model to have states $\mathfrak{T}/{\equiv}$ and with atomic action relations given by $[s] \xrightarrow{a} [t]$ iff $\exists s' \in [s], t' \in [t]. s' \xrightarrow{a} t'$. The number of equivalence classes is at most $2^{|\Gamma|}$ and so is $O(2^{|\phi|})$. The rest of the proof shows that the filtered model is indeed a model, in that $[s] \models \psi$ iff $s \models \psi$ for $\psi \in \Gamma$. For PDL the only case requiring comment is the case $\langle\alpha^*\rangle\psi$, which proceeds by an induction on the length of the witnessing sequence of α's. Consequently if ϕ is a satisfiable PDL formula, then it has a model with size $O(2^{|\phi|})$, and in fact $2^{|\phi|}$ suffices—see [25] for full details.

In order to obtain an upper bound for satisfiability from the small model property, we also need to know the complexity of model-checking, that is, determining whether $s \models \phi$. It is straightforward to define an inductive procedure for this, which is polynomial in the size of the formula and of the system. For example, to determine the truth of $\langle\alpha^*\rangle\phi$, one computes the *-closure of the α relation, and then checks for an α^*-successor satisfying ϕ. These results give an NTIME(c^n) (where c is a constant and n the formula size) upper bound for the satisfiability problem. By a reduction to alternating Turing machines, [25] also gave a lower bound of DTIME($c^{n/\lg n}$). A closer to optimal technique for satisfiability due to Pratt uses tableaux [58].

Although CTL, CTL* and $L\mu$ all have the finite model property, the filtration technique does not apply. If one filters $\mathfrak{T}$ through a finite set Γ containing $\forall\mathbf{F}Q$ unintended loops may be added. For instance if $\mathfrak{T}$ is $s_i \xrightarrow{a} s_{i+1}$ for $1 \leq i < n$ and Q is only true at state s_n then $s_i \models \forall\mathbf{F}Q$ for each i. But when n is large enough the filtered model will have at least one transition $[s_j] \xrightarrow{a} [s_i]$ when $i \leq j < n$, with the consequence that $[s_i] \not\models \forall\mathbf{F}Q$. The initial approach to showing the finite model property utilized semantic tableaux where one explicitly builds a model for a satisfiable formula which has small size. But such a technique is very particular, and more sophisticated methods based on automata are used for optimal results, as we shall mention later.

3 Syntax and semantics of modal mu-calculus

The defining feature of mu-calculi is the use of fixpoint operators. The use of fixpoint operators in program logics goes back at least to De Bakker, Park and Scott [56]. However, their use in modal logics of programs dates from work of Pratt, Emerson and Clarke and Kozen. Pratt's version [59] used a fixpoint operator like the minimization operator of recursion theory; although this is only superficially different, it seems to have dissuaded people from using the logic in that form. Emerson and Clarke added fixed points to a temporal logic to capture fairness and other correctness properties [21]. Kozen's [35] paper introduced $L\mu$ as we use it today, and established a number of basic results.

Fixpoint logics are traditionally considered hard to understand. Furthermore, their semantics requires a familiarity with material that, although not difficult, is often omitted from undergraduate mathematics or logic programmes. Whether for practical purposes, or to guide oneself through the formal proofs, it is therefore worthwhile to spend a little time on discussing an intuitive understanding of $L\mu$ before going on to the definitions.

3.1 Fixpoints as recursion

Suppose that $\mathfrak{S}$ is the state space of some system. For example $\mathfrak{S}$ could be the set of all processes reachable by arbitrary length sequences of transitions from some initial process. One way to provide semantics of a state-based modal logic is to map formulae ϕ to sets of states, that is to elements of $\wp\mathfrak{S}$. For any formula ϕ this mapping is given by $\|\phi\|$ [1] . The idea is that this mapping tells us at which states each formula holds. If we allow our logic to contain variables with interpretations ranging over $\wp\mathfrak{S}$, then we can view the semantics of a formula with a free variable, $\phi(Z)$, as a function $f\colon \wp\mathfrak{S} \to \wp\mathfrak{S}$. If $f(S) \subseteq f(S')$ whenever $S \subseteq S' \subseteq \mathfrak{S}$ then f is *monotonic*. If $f(S) = S$ then S is a *fixed point* of f (as repeated application of f leaves S unchanged). If we take the usual lattice structure on $\wp\mathfrak{S}$, given by set inclusion, and if f is a monotonic function, then by the Knaster–Tarski theorem we know that f has fixed points, and indeed has a unique maximal and a unique minimal fixed point. The maximal fixed-point is the union of *post-fixed points*, $\bigcup\{S \subset \mathfrak{S} \mid S \subseteq f(S)\}$, and the minimal fixed-point is the intersection of *pre-fixed points*, $\bigcap\{S \subset \mathfrak{S} \mid f(S) \subseteq S\}$. So we could extend our basic logic with a minimal fixpoint operator μ, so that $\mu Z.\phi(Z)$ is a formula whose semantics is the least fixed point of f; and similarly a maximal fixpoint operator ν, so that $\nu Z.\phi(Z)$ is a formula whose semantics is the greatest fixed point of f (when the semantics of $\phi(Z)$ is monotonic).

A good reason to do this is that it provides a semantics for *recursion*, and adding recursion to the usual modal logics provides a neat way of expressing all the usual operators of temporal logics. For example, consider the CTL formula $\forall\mathbf{G}\phi$, 'always ϕ'. Another way of expressing this is to say that it is a property X such that if X is true, then ϕ is true, and wherever we go next, X remains true; so X satisfies the modal implicational equation

$$X \Rightarrow \phi \wedge [-]X.$$

where $[-]X$ means that X is true at every immediate successor (see subsection 3.3). A solution to this equation is precisely a post-fixed point of the formula $\phi\wedge[-]X$. But which solution of the possibly many is appropriate? The only canonical post-fixed point is the largest, and this also makes sense, since if a state satisfies *some* solution, then it surely satisfies $\forall\mathbf{G}\phi$. Hence the meaning of the formula is the largest post-fixed point, which by standard theory is exactly the largest fixed point, $\nu X.\phi \wedge [-]Z$.

On the other hand, consider the CTL property $\exists\mathbf{F}\phi$, 'there exists a path on which ϕ eventually holds'. We could write this recursively as 'Y holds if either ϕ holds now, or there's some successor on which Y is true':

$$Y \Leftarrow \phi \vee \langle-\rangle Y.$$

Here we have a pre-fixed point of $\phi \vee \langle-\rangle Y$; the only canonical such is the least, and if a state satisfies $\exists\mathbf{F}\phi$, then it surely satisfies any solution Y' of the equation. Hence we want the least pre-fixed point, which is also the least fixed point, $\mu Y.\phi \vee \langle-\rangle Y$.

Finally, we observe that since we want the fixed points, we may replace the implications by equalities in the modal equations above, and get the same answers. It is therefore usual to cast modal fixpoint logics in terms of equations, rather than of implications.

3.2 Approximating fixpoints and μ as 'finitely'

The other key idea is that of approximants and unfolding. The standard theory tells us that if f is a monotonic function on a lattice, we can construct the least fixed point of f by applying f repeatedly on the bottom element $\bot$ of the lattice to form an increasing chain, whose limit is the fixed point. The length of the iteration is in general transfinite, but is bounded at worst by the cardinal after cardinality of the lattice, and in the special

[1] The mapping can be either given directly (inductively) or indirectly as the set $\{s \in \mathfrak{S} : s \models \Phi\}$.

case of a powerset lattice $\wp\mathfrak{S}$, by the cardinal after the cardinality of $\mathfrak{S}$. So if f is monotonic on $\wp\mathfrak{S}$, we have the increasing chain $\emptyset \subseteq f(\emptyset) \subseteq f^2(\emptyset) \subseteq \ldots \subseteq f^\alpha(\emptyset) \ldots$ and the least fixed point is the limit of this chain

$$\mu f = \bigcup_{\alpha<\kappa} f^\alpha(\emptyset)$$

and similarly as there is the anti-chain, $\mathfrak{S} \supseteq f(\mathfrak{S}) \supseteq f^2(\mathfrak{S}) \supseteq \ldots \supseteq f^\alpha(\mathfrak{S}) \ldots$,

$$\nu f = \bigcap_{\alpha<\kappa} f^\alpha(\mathfrak{S})$$

—or in terms of a infinitary logic, $\mu Z.\phi(Z) = \bigvee_{\alpha<\kappa} \phi^\alpha(\bot)$—where κ is at worst $|\mathfrak{S}| + 1$ for finite $\mathfrak{S}$, or $\aleph_1$ for countable $\mathfrak{S}$ (and $\nu Z.\phi(Z) = \bigwedge_{\alpha<\kappa} \phi^\alpha(\top)$). So for a minimal fixpoint $\mu Z.\phi(Z)$, if a state s satisfies the fixpoint, it satisfies some approximant, say for convenience the $\beta + 1$ th so that $s \models \phi^{\beta+1}(\bot)$. Now if we *unfold* this formula once, we get $s \models \phi(\phi^\beta(\bot))$. That is, the fact that s satisfies the fixpoint depends, via ϕ, on the fact that other states satisfy the fixpoint *at smaller approximants than s does*. So if one follows a chain of dependencies, the chain terminates. This is the strict meaning behind the slogan "μ means 'finite looping' ", which, with a little refinement, is sufficient to understand $L\mu$.

On the other hand, for a maximal fixpoint $\nu Z.\phi(Z)$, there is no such decreasing chain: $s \models \nu Z.\phi(Z)$ iff $s \models \phi(\nu Z.\phi(Z))$, and we may loop for ever, as in the process $P \overset{\text{def}}{=} a.P$, which repeatedly does an a action, and so satisfies $\nu Z.\langle a\rangle Z$. (However, if a state *fails* a maximal fixpoint, then there is a descending chain of failures.) Instead, we have the principle of fixpoint induction: if by assuming that a set $S \models Z$, we can show that $S \models \phi(Z)$, then we have shown that $S \models \nu Z.\phi$ (compare the recursive formulation of $\forall\mathbf{G}\phi$ in the previous section).

So in summary, one may understand fixpoints by the slogan 'ν means looping, and μ means finite looping'. This slogan provides an alternative means of explaining why a minimal fixpoint is required in the translation of $\exists\mathbf{F}\phi$. This formula means that there is a path on which ϕ eventually holds: that is, on the chosen path, ϕ holds within finite time. Hence the 'equation' $Y = \phi \vee \langle-\rangle Y$ must only be applied a finite number of times, and so by the slogan we should use a minimal fixpoint.

In the case of formulae with alternating fixpoints (which we shall examine a little later), the slogan remains valid, but requires a little more care in application. It is essential to almost all proofs about $L\mu$: the notion of 'well-founded premodel' with which Streett and Emerson [64] proved the finite model property, is an example of the slogan; so are the tableau model-checking approaches of Stirling and Walker [62], and Bradfield and Stirling [12].

3.3 Syntax of $L\mu$

Let Var be an (infinite) set of *variable names*, typically indicated by $Z, Y, \ldots$; let Prop be a set of *atomic propositions*, typically indicated by $P, Q, \ldots$; and let $\mathcal{L}$ be a set of *labels*, typically indicated by $a, b, \ldots$. The set of $L\mu$ formulae (with respect to $\mathrm{Var}, \mathrm{Prop}, \mathcal{L}$) is defined in parsimonious form as follows:

- P is a formula.
- Z is a formula.
- If ϕ_1 and ϕ_2 are formulae, so is $\phi_1 \wedge \phi_2$.
- If ϕ is a formula, so is $[a]\phi$.
- If ϕ is a formula, so is $\neg\phi$.
- If ϕ is a formula, then $\nu Z.\phi$ is a formula, provided that every free occurrence of Z in ϕ occurs positively, i.e. within the scope of an even number of negations. (The notions of free and bound variables are as usual, where ν is the only binding operator.)

If a formula is written as $\phi(Z)$, it is to be understood that the subsequent writing of $\phi(\psi)$ means ϕ with ψ substituted for all free occurrences of Z. There is no suggestion that Z is the only free variable of ϕ.

The positivity requirement on the fixpoint operator is a syntactic means of ensuring that $\phi(Z)$ denotes a functional monotonic in Z, and so has unique minimal and maximal fixpoint. It is usually more convenient to introduce derived operators defined by de Morgan duality, and work in positive form:

- $\phi_1 \vee \phi_2$ means $\neg(\neg\phi_1 \wedge \neg\phi_2)$.
- $\langle a \rangle \phi$ means $\neg[a]\neg\phi$.
- $\mu Z.\phi(Z)$ means $\neg\nu Z.\neg\phi(\neg Z)$.

Note the triple use of negation in μ, which is required to maintain the positivity. A formula is said to be in *positive form* if it is written with the derived operators so that $\neg$ only occurs applied to atomic propositions. It is in *positive normal form* if in addition all bound variables are distinct (and different from free variables). Any formula can be put into positive normal form by use of de Morgan laws and α-conversion. So we shall often assume positive normal form, and when doing structural induction on formulae will often take the derived operators as primitives.

For the concrete syntax, we shall assume that modal operators have higher precedence than boolean, and that fixpoint operators have lowest precedence, so that the scope of a fixpoint extends as far to the right as possible.

There are a few extensions to the syntax which are convenient in presenting examples, and in practice. The most useful is to allow modalities to refer not just to single actions, but to sets of actions. The most useful set is 'all actions except a'. So:

- $s \models [K]\phi$ iff $\forall a \in K. s \models [a]\phi$, and $[a, b, \ldots]\phi$ is short for $[\{a, b, \ldots\}]\phi$.
- $[-K]\phi$ means $[\mathcal{L} - K]\phi$, and set braces may be omitted.

Thus $[-]\phi$ means just $[\mathcal{L}]\phi$. [2]

3.4 *Semantics of $L\mu$*

An *$L\mu$ structure* $\mathfrak{T}$ (over $\mathrm{Prop}, \mathcal{L}$) is a labelled transition system, namely a set $\mathfrak{S}$ of states and a transition relation $\rightarrow\, \subseteq \mathfrak{S} \times \mathcal{L} \times \mathfrak{S}$ (as usual we write $s \xrightarrow{a} t$), together with an interpretation $\mathfrak{V}_{\mathrm{Prop}}\colon \mathrm{Prop} \rightarrow \wp\mathfrak{S}$ for the atomic propositions.

Given a structure $\mathfrak{T}$ and an interpretation $\mathfrak{V}\colon \mathrm{Var} \rightarrow \wp\mathfrak{S}$ of the variables, the set $\|\phi\|^{\mathfrak{T}}_{\mathfrak{V}}$ of states satisfying a formula ϕ is defined as follows:

$$\begin{aligned}
\|P\|^{\mathfrak{T}}_{\mathfrak{V}} &= \mathfrak{V}_{\mathrm{Prop}}(P) \\
\|Z\|^{\mathfrak{T}}_{\mathfrak{V}} &= \mathfrak{V}(Z) \\
\|\neg\phi\|^{\mathfrak{T}}_{\mathfrak{V}} &= \mathfrak{S} - \|\phi\|^{\mathfrak{T}}_{\mathfrak{V}} \\
\|\phi_1 \wedge \phi_2\|^{\mathfrak{T}}_{\mathfrak{V}} &= \|\phi_1\|^{\mathfrak{T}}_{\mathfrak{V}} \cap \|\phi_2\|^{\mathfrak{T}}_{\mathfrak{V}} \\
\|[a]\phi\|^{\mathfrak{T}}_{\mathfrak{V}} &= \{\, s \mid \forall t. s \xrightarrow{a} t \Rightarrow t \in \|\phi\|^{\mathfrak{T}}_{\mathfrak{V}} \,\} \\
\|\nu Z.\phi\|^{\mathfrak{T}}_{\mathfrak{V}} &= \bigcup \{ S \subseteq \mathfrak{S} \mid S \subseteq \|\phi\|^{\mathfrak{T}}_{\mathfrak{V}[Z:=S]} \}
\end{aligned}$$

where $\mathfrak{V}[Z := S]$ is the valuation which maps Z to S and otherwise agrees with $\mathfrak{V}$. If we are working in positive normal form, we may add definitions for the derived operators by duality:

$$\begin{aligned}
\|\phi_1 \vee \phi_2\|^{\mathfrak{T}}_{\mathfrak{V}} &= \|\phi_1\|^{\mathfrak{T}}_{\mathfrak{V}} \cup \|\phi_2\|^{\mathfrak{T}}_{\mathfrak{V}} \\
\|\langle a \rangle\phi\|^{\mathfrak{T}}_{\mathfrak{V}} &= \{\, s \mid \exists t. s \xrightarrow{a} t \wedge t \in \|\phi\|^{\mathfrak{T}}_{\mathfrak{V}} \,\} \\
\|\mu Z.\phi\|^{\mathfrak{T}}_{\mathfrak{V}} &= \bigcap \{ S \subseteq \mathfrak{S} \mid S \supseteq \|\phi\|^{\mathfrak{T}}_{\mathfrak{V}[Z:=S]} \}
\end{aligned}$$

[2] Beware that many authors use '$[\,]\phi$' to mean '$[\mathcal{L}]\phi$', rather than the (vacuous) '$[\emptyset]\phi$' that it means in our notation.

We drop $^{\mathfrak{T}}$ and $_{\mathfrak{V}}$ whenever possible; and write $s \models \phi$ for $s \in \|\phi\|$.

We have discussed informally the importance of approximants; let us now define them. If $\mu Z.\phi(Z)$ is a formula, then for an ordinal α, let $\mu Z^{\alpha}.\phi$ and $\mu Z^{<\alpha}.\phi$ be formulae, with semantics given, with simultaneous induction on α, by:

$$\begin{aligned}
\|\mu Z^{<\alpha}.\phi\|^{\mathfrak{T}}_{\mathfrak{V}} &= \bigcup_{\beta<\alpha} \|\mu Z^{\beta}.\phi\|^{\mathfrak{T}}_{\mathfrak{V}} \\
\|\mu Z^{\alpha}.\phi\|^{\mathfrak{T}}_{\mathfrak{V}} &= \|\phi\|^{\mathfrak{T}}_{\mathfrak{V}[Z:=\|\mu Z^{<\alpha}.\phi\|^{\mathfrak{T}}_{\mathfrak{V}}]}
\end{aligned}$$

The approximants of a maximal fixpoint are defined dually:

$$\begin{aligned}
\|\nu Z^{<\alpha}.\phi\|^{\mathfrak{T}}_{\mathfrak{V}} &= \bigcap_{\beta<\alpha} \|\nu Z^{\beta}.\phi\|^{\mathfrak{T}}_{\mathfrak{V}} \\
\|\nu Z^{\alpha}.\phi\|^{\mathfrak{T}}_{\mathfrak{V}} &= \|\phi\|^{\mathfrak{T}}_{\mathfrak{V}[Z:=\|\nu Z^{<\alpha}.\phi\|^{\mathfrak{T}}_{\mathfrak{V}}]}
\end{aligned}$$

Note that $\mu Z^{<0}.\phi \Leftrightarrow \bot$ and $\nu Z^{<0}.\phi \Leftrightarrow \top$. By abuse of notation, we write Z^{α} or ϕ^{α} to mean ${}^{\mu}_{\nu} Z^{\alpha}.\,\phi$; of course this only makes sense when one knows which fixpoint and variable is meant.

We should remark here that most literature on $L\mu$ uses a slightly different definition, putting $\mu Z^{0}.\phi = \bot$, $\mu Z^{\alpha+1}.\phi = \phi(\mu Z^{\alpha}.\phi)$, and $\mu Z^{\lambda}.\phi = \bigcup_{\beta<\lambda} \mu Z^{\beta}.\phi$ for limit λ—which in effect is writing α for our $<\alpha$. That notation is taken from set theory; its advantage is that a limit approximant is the limit of approximants. Our notation is taken from more recent set theoretic practice; its advantages are that it sometimes reduces the number of trivial case distinctions in inductive proofs. However, the difference is not significant.

Sometimes, we are interested in *rooted* structures $(\mathfrak{T}, s_0, \mathfrak{V}_{\mathrm{Prop}})$ for $L\mu$ formulae that have a designated initial state s_0: ϕ is true of such a structure if $s_0 \models \phi$. We can, therefore, examine the set of all rooted structures where ϕ is true which allows comparison between $L\mu$ and other notations for classifying structures.

3.5 Examples

We have seen, both informally and in the formal semantics, the meaning of the fixpoint operators, and we have seen some simple examples of $L\mu$ translating CTL. We now consider some examples of $L\mu$ formulae in their own right, which express properties one might meet in practice.

There is a well-known ‘classification’ [42] of basic properties into safety and liveness. In terms of $L\mu$, it is not unreasonable to say that μ is liveness and ν is safety. Consider first simple ν formulae. For example:

$$\nu Z.P \wedge [a]Z$$

is a relativized ‘always’ formula: ‘P is true along every a-path’. Slightly more complex is the relativized ‘while’ formula

$$\nu Z.Q \vee (P \wedge [a]Z)$$

‘on every a-path, P holds while Q fails’. Both formulae can be understood directly via the fixpoint construction, or via the idea of ‘ν as looping’: for example the second formula is true if either Q holds, or if P holds and wherever we go next (via a), the formula is true, and ..., and because the fixpoint is maximal, we can repeat forever. So in particular, if P is always true, and Q never holds, the formula is true.

μ formulae, in contrast, require something to happen, and thus are liveness properties. For example

$$\mu Z.P \vee [a]Z$$

is ‘on all infinite length a-paths, P eventually holds’; and

$$\mu Z.Q \vee (P \wedge \langle a\rangle Z)$$

is ‘on some a-path, P holds until Q holds (and Q *does* eventually hold)’. Again, these can be understood by ‘μ as finite looping’: in the second case, we are no longer allowed to repeat the unfolding forever, so we must eventually ‘bottom out’ in the Q disjunct.

This level of complexity suffices to translate CTL, since we have $\mu Z.Q \vee (P \wedge \langle - \rangle Z)$ as a translation of $\exists[P\ \mathbf{U}\ Q]$, and $\mu Z.Q \vee (P \wedge [-]Z \wedge \langle - \rangle \top)$ as a translation of $\forall[P\ \mathbf{U}\ Q]$ (the conjunct $\langle - \rangle \top$ ensures that Q is actually reached, since $[-]Z$ is true at deadlock states); and obviously we can nest formulae inside one another, such as

$$\nu Z.(\mu Y.P \vee \langle - \rangle Y) \wedge [-]Z$$

'it is always possible that P will hold', or $\forall \mathbf{G}(\exists \mathbf{F} P)$. Equally obviously, we can write formulae with no CTL translation, such as

$$\mu Z.[a]\bot \vee \langle a \rangle \langle a \rangle Z$$

which asserts the existence of a maximal a-path of even length; a formula which is, incidentally, expressible in PDL. This is, however, a fairly simple extension; much more interesting is the power one gets from mixing fixpoints that depend on one another. Consider the formula

$$\mu Y.\nu Z.(P \wedge [a]Y) \vee (\neg P \wedge [a]Z).$$

This formula usually gives pause for thought, but it has a simple meaning, which can be seen by using the slogans. $\mu Y. \ldots$ is true if $\nu Z. \ldots$ is true if $(P \wedge [a]Y) \vee (\neg P \wedge [a]Z)$, which is true if either P holds and at the next (a)-states we loop back to $\mu Y. \ldots$, or P fails, and at the next states we loop back to $\nu Z. \ldots$. By the slogan 'μ means finitely', we can only loop through $\mu Y. \ldots$ finitely many times on any path, and hence P is true only finitely often on any path.

We shall see in a later section that this so-called alternation of fixpoint operators does indeed give ever more expressive power as the number of alternations increases. It also appears to increase the complexity of model-checking: all known algorithms are exponential in the alternation, but whether this is necessarily the case is the main remaining open problem about $L\mu$.

3.6 Fixpoint regeneration and the 'fundamental semantic theorem'

In the informal description of the meaning of fixpoints, we used the idea of the dependency of s at ϕ on t at ψ. We now make this precise. Assume a structure $\mathfrak{T}$, and a formula ϕ. Suppose that we annotate the states with sets of subformulae, such that the sets are locally consistent: that is, s is annotated with a conjunction iff it is annotated with both conjuncts; s is annotated with a disjunction iff it is annotated with at least one disjunct; if s is annotated with $[a]\psi$ (resp. $\langle a \rangle \psi$), then each (resp. at least one) a-successor is annotated with ψ; if s is annotated with a fixpoint or fixpoint variable, it is annotated with the body of the fixpoint. We call such an annotated structure a *quasi-model*.

A *choice function* f is a function which for every disjunctive subformula $\psi_1 \vee \psi_2$ and every state annotated with $\psi_1 \vee \psi_2$ chooses one disjunct $f(s, \psi_1 \vee \psi_2)$; and for every subformula $\langle a \rangle \psi$ and every state s annotated with $\langle a \rangle \psi$ chooses one a-successor $t = f(s, \langle a \rangle \psi)$ annotated with ψ.

A *pre-model* is a quasi-model equipped with a choice function.

Given a pre-model with choice function f, the *dependencies* of a state s that satisfies a subformula ψ are defined thus: $s@\psi_1 \wedge \psi_2 \succ s@\psi_i$ for $i = 1, 2$; $s@[a]\psi \succ t@\psi$ for every t such that $s \xrightarrow{a} t$; $s@\psi_1 \vee \psi_2 \succ s@f(s, \psi_1 \vee \psi_2)$; $s@\langle a \rangle \psi \succ f(s, \langle a \rangle \psi)@\psi$; $s@{}^{\mu}_{\nu}Z.\,\psi \succ s@\psi$; $s@Z \succ s@\psi$ where Z is bound by ${}^{\mu}_{\nu}Z.\,\psi$. A *trail* is a maximal chain of dependencies.

If every trail has the property that the highest (i.e. with the outermost binding fixpoint) variable occurring infinitely often is a ν-variable, the pre-model is *well-founded*. (Equivalently: in any trail, a μ-variable can only occur finitely often unless a higher variable is encountered.)

The fundamental theorem on the semantics of $L\mu$ can now be stated:

Theorem 3.1 *A well-founded pre-model is a model: in a well-founded pre-model, if s is annotated with ψ, then indeed $s \models \psi$.*

The theorem in this form is due to Streett and Emerson in [64], from which the term 'well-founded pre-model' is taken. Stirling and Walker [63] presented a tableau system for model-checking on finite structures, and the soundness theorem for that system is essentially a finite version of the fundamental theorem using a more relaxed notion of choice; the later infinite-state version of [12,7] is the fundamental theorem, again with a slight relaxation on choice.

A converse is also true:

Theorem 3.2 *If in some structure $s \models \phi$, then there is a* locally consistent *annotation of the structure and a choice function which make the structure a well-founded pre-model.*

The fundamental theorem, in its various guises, is the precise statement of the slogan 'μ means finite looping'. To explain why it is true, and to define the term 'locally consistent', we need to make a finer analysis of approximants.

Assume a structure $\mathfrak{T}$, valuation $\mathfrak{V}$, and formula ϕ in positive normal form. Let $Y_1, \ldots, Y_n$ be the μ-variables of ϕ, in an order compatible with formula inclusion: that is, if $\mu Y_j.\psi_j$ is a subformula of $\mu Y_i.\psi_i$, then $i \leq j$. If Y_i is some inner fixpoint, then its denotation depends on the meaning of the fixpoints enclosing it: for example, in the formula $\mu Y_1.\langle a \rangle \mu Y_2.(P \vee Y_1) \vee \langle b \rangle Y_2$, to calculate the inner fixpoint μY_2 we need to know the denotation of Y_1. We may ask: what is the least approximant of Y_1 that could be plugged in to make the formula true? Having fixed that, we can then ask what approximant of Y_2 is required. This idea is the notion of *signature*. A signature is a sequence $\sigma = \alpha_1, \ldots, \alpha_n$ of ordinals, such that the i least fixpoint will be interpreted by its α_ith approximant (calculated relative to the outer approximants).

The definition and use of signatures inevitably involves some slightly irritating book-keeping, and they appear in several forms in the literature. In [64], the Fischer–Ladner closure of ϕ was used, rather than the set of subformulae. The closure is defined by starting with ϕ and closing under the operations of taking the immediate components of formulae with boolean or modal top-level connectives, together with the rule that if ${}^{\mu}_{\nu}Z.\,\psi(Z) \in \mathrm{cl}(\phi)$, then $\psi({}^{\mu}_{\nu}Z.\,\psi) \in \mathrm{cl}(\phi)$. The signatures were defined by syntactically unfolding fixpoints, rather than by semantic approximants. In [63] and following work, a notion of *constant* was used, which allows some of the book-keeping to be moved into the logic. Although all the notions and proofs using them are interconvertible, the 'constant' variant is perhaps easier to follow, and has the advantage that it adapts easily to the modal equation system presentation of $L\mu$, which we shall see below. Indeed, it arises more naturally from that system.

Add to the language a countable set of *constants* $U, V, \ldots$. Constants will be defined to stand for maximal fixpoints or approximants of minimal fixpoints. Specifically, given a formula ϕ, let $Y_1, \ldots, Y_n$ be the μ-variables as above, let $Z_1, \ldots, Z_m$ be the ν-variables, let $\sigma = \alpha_1, \ldots, \alpha_n$ be a signature, and let $U_1, \ldots, U_n, V_1, \ldots, V_m$ be constants, which will be associated with the corresponding variables. They are given semantics thus: if Y_i is bound by $\mu Y_i.\psi_i$, then $\|U_i\|_\sigma$ is $\|\mu Y_i^{\alpha_i}.\psi_i'\|_\sigma$, where ψ_i' is obtained from ψ_i by substituting the corresponding constants for the free fixpoint variables of $\mu Y_i.\psi_i$. If Z_i is bound by $\nu Z_i.\psi_i$, its semantics is $\|\nu Z_i.\psi_i'\|_\sigma$. Given an arbitrary subformula ψ of ϕ, we say a state s satisfies ψ with signature σ, written $s \models_\sigma \psi$, if $s \in \|\psi'\|_\sigma$, where ψ' is ψ with its free fixpoint variables substituted by the corresponding constants.

Order signatures lexicographically. Now, given a pre-model for ϕ, extend the annotations so that each subformula at s is accompanied by a signature – write $s@\psi[\sigma]$. Such an extended annotation is said to be locally consistent if the signature is unchanged or decreases by passing through boolean, modal, or ν-variable dependencies, and when passing through $s@Y_i$ it strictly decreases in the ith component and is unchanged in the $1, \ldots, i-1$'th components.

It can now be shown, by a slightly delicate but not too difficult induction on ψ and σ, that if $s@\psi[\sigma]$, then $s \models_\sigma \psi$. The proof proceeds by contradiction: suppose that $s@\psi[\sigma]$ and $s \not\models_\sigma \psi$. If ψ is $\psi_1 \vee \psi_2$ ($\psi_1 \wedge \psi_2$) then for some $i \in \{1, 2\}$, $s@\psi_1[\sigma]$ and $s \not\models_\sigma \psi_i$. If ψ is $[a]\psi'$ ($\langle a \rangle \psi'$) then for some s', $s \xrightarrow{a} s'$, $s'@\psi'[\sigma]$ and $s' \not\models_\sigma \psi'$. If ψ is a least fixpoint variable Y_i, then we pass through the unfolding rule to $s@\psi_i[\sigma']$ where $\sigma' < \sigma$ and $s \not\models_{\sigma'} \psi_i$. (Hence we can only pass through least fixpoints finitely often.) The key case is when ψ

is a greatest fixpoint variable Z_i. In this case, the hypothesis is that $s@Z_i[\sigma]$ and $s \not\models_\sigma Z_i$. Then s fails some approximant Z_i^β (and $s@Z_i^\beta[\sigma]$); and then passing through the unfolding rule gives s fails $\psi_i^{\beta'}$ for $\beta' < \beta$ (and $s@\psi_i^{\beta'}[\sigma]$). Continuing to chase the falsehood down the pre-model, we eventually arrive at a state failing the zero'th approximant of a greatest fixpoint formula, which is impossible.

Furthermore, given a well-founded pre-model, one can construct a locally consistent signature annotation—essentially, the Y_i component of σ in $s@\psi[\sigma]$ is the maximum 'number' (in the transfinite sense) of Y_i occurrences without meeting a higher variable in trails from $s@\psi$, and so on; the well-foundedness of the pre-model guarantees that this is well-defined. This gives the fundamental theorem.

The converse is quite easy: given a model, annotate the states by the subformulae they satisfy; for $s@\psi$ assign the least σ such that $s \models_\sigma \psi$; and choose a choice function that always chooses the successor with least signature. It is easy to show that this is a well-founded pre-model and signature assignment.

3.7 *Modal equation systems*

The presentation of $L\mu$ so far is a traditional logical formulation. However, in several circumstances it can be useful to think in terms of systems of recursive equations for the fixpoint variables, as follows.

A *modal equation system* comprises a sequence $B_0; \ldots ; B_n$ of *blocks*; each B_i may be a μ*-block* (we write B_i^μ) or a ν*-block* (we write B_i^ν). Each block $B_i^{\mu/\nu}$ is a sequence of equations $X_{i0} \overset{\mu/\nu}{=} \phi_{i0},\ \ldots,\ X_{ik_i} \overset{\mu/\nu}{=} \phi_{ik_i}$, where each ϕ_{ij} is a modal formula which may contain any of the variables $X_{i'j'}$ positively.

Thus each block B_i defines a functional on vectors $(S_{i0}, \ldots, S_{ik_i}) \in (\wp\mathfrak{S})^{k_i}$. This functional is relative to valuations of the variables in earlier blocks, and refers to the solutions of later blocks. If B_i^μ, then take the least fixpoint (in the componentwise ordering) of this functional, and if B_i^ν, take the greatest. Conventionally, the solution of the entire equation system is taken to be the solution for the first variable X_{00}.

There is an obvious transformations from $L\mu$ to modal equation systems: for example, $\mu X.P \vee \nu Y.[a]Y \wedge [b]X$ translates to

$$X_{00} \overset{\mu}{=} P \vee X_{10} \quad ; \quad X_{10} \overset{\nu}{=} [a]X_{10} \wedge [b]X_{00}.$$

Similarly, there is a reasonably obvious reverse transformation: for example, the equation system

$$X_{00} \overset{\mu}{=} \langle a\rangle X_{10} \vee [b]X_{10} \quad ; \quad X_{10} \overset{\nu}{=} P \wedge [a](X_{00} \vee X_{10})$$

translates to $\mu X.\langle a\rangle(\nu Y.P \wedge [a](X \vee Y)) \vee [b](\nu Y.P \wedge [a](X \vee Y))$. These translations, known from finite model theory, show that modal equation systems and $L\mu$ are equi-expressive. Note that in the second example, the formula duplicates the second equation: by extending such examples, one can see that the translation from equation systems to formulae may introduce an exponential blow-up. However, this blow-up results in formulae with many identical sub-formulae, which can in any case be optimized away during model-checking, and in general problems in modal equation systems are of the same complexity as in $L\mu$.

A block in a modal equation system is to be understood as a *simultaneous* fixpoint. $L\mu$ could be directly presented with simultaneous fixed points: for instance, $s \models \mu Z_1 \ldots Z_n.(\phi_1, \ldots, \phi_n)$ iff $s \in S_1$ where $(S_1, \ldots, S_n) = \bigcap \{(S'_1, \ldots, S'_n) \mid S'_j \supseteq \|\phi_j\|^{\mathfrak{T}}_{\mathfrak{V}[Z_1:=S'_1,\ldots,Z_n:=S'_n]}\}$.

One of the main applications of modal equation systems is in the development of fast model-checking algorithms: modal equation systems can be easily translated to *boolean equation systems* (defined as above, but with boolean variables and just propositional equations) by having one boolean variable for each (modal variable, state) pair. Then graph-theoretic or matrix-theoretic techniques can be employed to solve the boolean equation systems. For more on this topic, see [46].

4 Expressive power

As we noted earlier in this article, there are many temporal logics used in practice, some of which are also historical precursors to $L\mu$. We said that most of them could be seen as fragments of $L\mu$. In this section we consider questions of expressivity and related topics, and start by showing how a number of other logics can be translated into $L\mu$.

4.1 CTL and friends as fragments of $L\mu$

PDL can be easily translated into $L\mu$ by unpacking the modal operators $\langle\alpha\rangle$: $\langle\alpha \cup \beta\rangle\psi = \langle\alpha\rangle\psi \vee \langle\beta\rangle\psi$, $\langle\alpha;\beta\rangle\psi = \langle\alpha\rangle\langle\beta\rangle\psi$ and $\langle\alpha^*\rangle\psi = \mu Z.\psi \vee \langle\alpha\rangle Z$. The logic CTL is one of the simplest temporal logics, and its translation is also simple. Recall the syntax and semantics of CTL from 2.2. The two basic operators are $\forall[\phi\ \mathbf{U}\ \psi]$ and $\exists[\phi\ \mathbf{U}\ \psi]$. Assuming that there are no deadlocked states, these can be simply translated as:

$$\mu Z.\psi \vee [-]\phi \qquad \text{and} \qquad \mu Z.\psi \vee \langle-\rangle\phi$$

with the proof of the equivalence being a straightforward application of the semantics. For both PDL and CTL, only a fragment of $L\mu$ is necessary where there is no essential alternation of fixpoints (as described in 7).

A much less trivial case is the logic CTL*. CTL* is the logic obtained by removing the syntactic constraint of CTL that requires every $\mathbf{U}$ to be immediately quantified by $\forall$ or $\exists$, so that in CTL* one can write formulae such as $\forall[(\phi\,\mathbf{U}\,\psi) \vee \neg(\phi'\,\mathbf{U}\,\psi')]$. Consequently, not all CTL* formulae have meanings purely in terms of states, and the question of translation into a purely state-based logic like $L\mu$ becomes problematic. However, one can ask the question, is every state formula of CTL* (that is, boolean combinations of atoms and quantified formulae) equivalent to an $L\mu$ formula? The answer is ‘yes’, but it is a harder problem. Wolper, in an unpublished note from the early 1980s, noted that state formulas of CTL* can be translated via automata theory into PDL over a single label with looping (which, in turn, is directly translatable into $L\mu$). The first explicit translation was given by Dam [17], but the translation is very difficult, and gives a doubly exponential blowup in the formula size. The latter means that the translation is of no use for model-checking, as existing CTL* algorithms are much faster than a double exponential blowup of $L\mu$ model-checking. A few years later, Bhat and Cleaveland [6] gave a single exponential translation into the equational variant of $L\mu$. Although still quite complex, utilising a so-called Büchi tableau automaton as an intermediary, this translation is efficient enough to give competitive model-checking of CTL* via translation.

4.2 Bisimulation and tree model property

Bisimulation or back-and-forth equivalence or zig-zag equivalence is the equivalence associated with modal logic. In our setting, a *bisimulation* between two $L\mu$ structures $\mathfrak{T}_1$ and $\mathfrak{T}_2$ over the same proposition set Prop and label set $\mathcal{L}$ is a relation R such that for all propositions P, if $P(s_1)$ and $s_1 R s_2$, then $P(s_2)$, and conversely; and if $s_1 R s_2$, and $s_1 \xrightarrow{a} s_1'$, then for some s_2', $s_2 \xrightarrow{a} s_2'$ and $s_1' R s_2'$, and conversely. Two states s_1 and s_2 are *bisimilar* if there is some bisimulation R such that $s_1 R s_2$.

Recall that HML is the fixpoint-free part of $L\mu$. The following is easily shown by structural induction on formulae:

Theorem 4.1 *If two states (in the same or different systems) are bisimilar, they satisfy the same HML formulae.*

By an induction on approximants, it is also straightforward to extend this to

Theorem 4.2 *If two states (in the same or different systems) are bisimilar, they satisfy the same $L\mu$ formulae.*

A system is *image-finite* if for all states s and labels a, the set $\{\, s' \mid s \xrightarrow{a} s' \,\}$ is finite. The following theorem holds:

Theorem 4.3 *If two states in image-finite systems satisfy the same HML (or $L\mu$) formulae, then they are bisimilar.*

To prove this, one observes that bisimulation itself is a maximal fixpoint, namely the maximal fixpoint of the map $R \mapsto \{\, (s_1, s_2) \mid (s_1 \xrightarrow{a} s_1' \Rightarrow \exists s_2'.s_2 \xrightarrow{a} s_2' \wedge (s_1', s_2') \in R) \wedge (s_2 \xrightarrow{a} s_2' \Rightarrow \exists s_1'.s_1 \xrightarrow{a} s_1' \wedge (s_1', s_2') \in$

$R) \}$ (ignoring the propositions, which can be dealt with by an additional clause); shows that in an image-finite system the approximants to this fixpoint close at ω; and then deduce that if two states are not bisimilar, there is a finite modal formula distinguishing them. The latter theorem does not hold for general systems: there are systems which satisfy the same $L\mu$ formulae, but are not bisimilar. The following example is based on one by Roope Kaivola. Let $\phi_1, \phi_2, \ldots$ be an enumeration of all $L\mu$ formulae over some finite label set $\mathcal{L}$. Let $\mathfrak{T}_i$, with initial state s_i, be a finite model for ϕ_i, with all $\mathfrak{T}_i$ disjoint. Let $\mathfrak{T}_0$ be constructed by taking an initial state s_0 and making $s_0 \xrightarrow{a} s_i$ for all $i > 0$. Let $\mathfrak{T}'_0$ be $\mathfrak{T}_0$ with the addition of a transition $s_0 \xrightarrow{a} s_0$. $\mathfrak{T}_0$ and $\mathfrak{T}'_0$ are clearly not bisimilar, because in $\mathfrak{T}'_0$ it is possible to defer indefinitely the choice of which $\mathfrak{T}_i$ to enter. On the other hand, suppose that ψ is a formula, and w.l.o.g. assume the topmost operator is a modality. If the modality is $[b]$, ψ is true of both $\mathfrak{T}_0$ and $\mathfrak{T}'_0$; if it is $\langle b \rangle$, ψ is false of both; if ψ is $\langle a \rangle \psi'$, then ψ is false at both $\mathfrak{T}_0$ and $\mathfrak{T}'_0$ iff ψ' is unsatisfiable, and true at both otherwise; if ψ is $[a]\psi'$, then ψ is true at both $\mathfrak{T}_0$ and $\mathfrak{T}'_0$ iff ψ' is valid, and false at both otherwise.

A simple corollary of theorem 4.2 is that $L\mu$ has the *tree model property*.

Proposition 4.4 *If a formula has a model, it has a model that is a tree.*

Just unravel the original model, thereby preserving bisimulation. This can be strengthened to the *bounded branching degree* tree model property (just cut off all the branches that are not actually required by some diamond subformula; this leaves at most (number of diamond subformulae) branches at each node).

4.3 $L\mu$ and automata

$L\mu$ is intimately related to automata theory, and the equivalence between various automata, particularly alternating parity automata, as described in section 5, and $L\mu$ is a key technique in many results. The first connexion between $L\mu$ and automata was tree automata, which we now briefly review.

Let us start with the notion of an automaton familiar from introductory computer science courses. A finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ consists of a finite set of states Q, a finite alphabet Σ, a transition function δ, an initial state $q_0 \in Q$ and an acceptance condition F. Classical nondeterministic automata recognise languages, subsets of Σ^*, where the transition function $\delta : Q \times \Sigma \rightarrow \wp Q$. Given a word $w = a_0 \ldots a_n \in \Sigma^*$, a *run* of $\mathcal{A}$ on w is a sequence of states $q_0 \ldots q_n$ that traverses w, so $q_{i+1} \in \delta(q_i, a_{i+1})$. The run is *accepting* if the sequence $q_0 \ldots q_n$ obeys F: classically, $F \subseteq Q$ and $q_0 \ldots q_n$ is accepting if the last state $q_n \in F$. There may be many different runs of $\mathcal{A}$ on w, some accepting the others rejecting, or no runs at all. The language *recognised* by $\mathcal{A}$ is the set of words for which there is at least one accepting run. A simple extension is to allow recognition of infinite length words. A run of $\mathcal{A}$ on $w = a_1 \ldots a_i \ldots$ is an infinite sequence of states $\pi = q_0 \ldots q_i \ldots$ that travels over w, so $q_{i+1} \in \delta(q_i, a_{i+1})$ and it is accepting if it obeys the condition F. Let *inf*$(\pi) \subseteq Q$ contain exactly the states that occur infinitely often in π. Classically, $F \subseteq Q$ and π is accepting if *inf*$(\pi) \cap F \neq \emptyset$ which is the Büchi acceptance condition.

Büchi automata are an alternative notation for characterizing infinite runs of systems. Assume Prop is a finite set. The alphabet $\Sigma = \wp\mathrm{Prop}$. If $\pi = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots$ is an infinite run, then $\pi \models \mathcal{A}$ if the automaton accepts the word $\mathrm{Prop}(s_0)\,\mathrm{Prop}(s_1) \ldots$ where $\mathrm{Prop}(s_i)$ is the subset of Prop that is true at s_i. For example, if $\mathrm{Prop} = \{P\}$, $Q = \{p, q\}$, $\delta(p, \{P\}) = \{q\}$, $\delta(p, \emptyset) = \{p\}$, $\delta(q, \{P\}) = \{q\}$ and $\delta(q, \emptyset) = \{q\}$, $q_0 = p$ and $F = \{q\}$, then this automaton is equivalent to the temporal formula $\mathbf{F}P$. (In fact, Büchi automata coincide with the *linear-time μ-calculus* where fixpoints are added to simple next time temporal logic that has the sole modality $\bigcirc$.)

When formulae are equivalent to automata, satisfiability checking reduces to the *non-emptiness* problem for the automata: that is, whether the automaton accepts something. If $\mathcal{A}$ is a Büchi automaton, then it is non-empty if there is a transition path $q_0 \longrightarrow^* q \in F$ and a cycle $q \longrightarrow^* q$ (equivalent to an eventually cyclic model).

The notion of bounded branching *tree automaton* extends the definition of automaton to accept n-branching infinite trees whose nodes are labelled with elements of Σ. Previously, states q' belonged to $\delta(q, a)$; now it is tuples $(q'_1, \ldots, q'_n)$ that belong to $\delta(q, a)$. A tree automaton traverses the tree, descending from a node to all n-child nodes, so the automaton splits itself into n copies, and proceeds independently. A run of the automaton is then an n-branching infinite tree labelled with states of the automaton. A run is accepting if *every* path through this tree satisfies the acceptance condition F. In the case of Rabin acceptance $F = \{(G_1, R_1), \ldots (G_k, R_k)\}$ where each $G_i, R_i \subseteq Q$ and π obeys F if there is a j such that $\mathit{inf}(\pi) \cap G_j \neq \emptyset$ and $\mathit{inf}(\pi) \cap R_j = \emptyset$. A variant definition is *parity* acceptance first introduced (not under that name) by Mostowski [52] where F maps each state q of the automaton to a *priority* $F(q) \in \mathbb{N}$. We say that a path satisfies F if the least priority seen infinitely often is even. It is not hard to see that a parity condition is a special case of a Rabin condition; it is also true, though somewhat trickier, that a Rabin automaton can be translated to an equivalent parity automaton.

Assuming Prop is finite, tree automata characterize rooted n-branching infinite tree models $(\mathfrak{T}, s_0, \mathfrak{V}_{\mathrm{Prop}})$ for $L\mu$ formulae where s_0 is the root of the tree: $(\mathfrak{T}, s_0, \mathfrak{V}_{\mathrm{Prop}}) \models \mathcal{A}$ if $\mathcal{A}$ accepts the behaviour tree $\mathfrak{T}'$ that replaces each state $s \in \mathfrak{T}$ with $\mathrm{Prop}(s)$. For example, let $\mathrm{Prop} = \{P\}$, $Q = \{p, q\}$, $\delta(p, \{P\}) = \{(p, p)\}$, $\delta(p, \emptyset) = \{(q, q)\}$, $\delta(q, \{P\}) = \{(p, p)\}$ and $\delta(q, \emptyset) = \{(q, q)\}$ and $q_0 = p$. This automaton $\mathcal{A}$ with parity acceptance $F(p) = 1$ and $F(q) = 2$ is equivalent to $\mu Y.\nu Z.(P \wedge [a]Y) \vee (\neg P \wedge [a]Z)$ over infinite binary-tree models: the fixed point μY is ‘coded’ by p and νZ is coded by q.

The use of priorities looks very much like the definition of well-founded pre-model from section 3.6, if we assign priorities to the subformulae of an $L\mu$ formula in such a way that the priority of a fixpoint formula is lower than any of its subformulae (and the priority of a least fixpoint is odd). Indeed, it is essentially the same thing. Tree automata and $L\mu$ are equivalent [64]:

Theorem 4.5 *A family of n-branching infinite tree models is defined by some tree automaton iff it is the set of n-branching infinite tree models of some corresponding $L\mu$ formula. Consequently, determining whether a system satisfies an $L\mu$ formula is equivalent to determining whether its behaviour trees are accepted by the corresponding automaton.*

Decidability of satisfiability of $L\mu$ formulae reduces to the non-emptiness problem for tree automata. This problem is more difficult than for Büchi automata. However, there is an exponential decision procedure that is inductive in the *index* of the automaton (which is the number of parities or pairs in F, in the case of a Rabin automaton).

This illustrates the potency of the automata-theoretic approach to temporal logic that has become popular in recent years. Satisfiability of formulae is reduced to the non-emptiness problem for a class of automata. There is also the virtue that automata sustain combinatorial transformations, such as determinization, and closure operations, such as intersection, that are not in the logical repertoire. Occasionally, logics are easier: one of the hardest automata-theoretic proofs is that tree automata are closed under complementation. We shall see more of automata in later sections.

4.4 $L\mu$ and games

$L\mu$ is also intimately related to games, as are automata. We can view the relationship at different levels.

The fundamental semantic theorem can be presented as a simple two player *model-checking* game. Assume a rooted model $(\mathfrak{T}, s_0, \mathfrak{V})$ and formula ϕ_0 in positive normal form. The game $G(s_0, \phi_0)$ is defined on an *arena* that is a set of pairs (s, ψ) where s is a state of $\mathfrak{T}$ and ψ is a subformula of ϕ_0. The initial position is (s_0, ϕ_0). There are two players, whom we will call simply $\forall$ and $\exists$. (Other popular names include Player II/Player I, Abelard/Eloise, Opponent/Proponent, Refuter/Verifier.) $\forall$ is responsible for making a move from a position $(s, \phi \wedge \psi)$, the available choices are $\{(s, \phi), (s, \psi)\}$, and from a position $(s, [a]\phi)$ whose available choices are $\{(t, \phi) \mid s \xrightarrow{a} t \in \mathfrak{T}\}$. Similarly, $\exists$ is responsible for $(s, \phi \vee \psi)$ and $(s, \langle a \rangle \phi)$. There are final positions (s, ψ) where $\psi \in \{P, \neg P, [a]\phi, \langle a \rangle \phi\}$: $(s, [a]\psi)$ and $(s, \langle a \rangle \phi)$ are only final if there is no state t such that $s \xrightarrow{a} t$. A final position (s, ψ) is winning for $\exists$ if $s \models \psi$; otherwise it is winning for $\forall$.

A play of $G(s_0, \phi_0)$ is a finite or infinite sequence of positions starting with (s_0, ϕ_0). $\exists$ wins a finite play if the final position is winning for $\exists$. She wins an infinite play if the outermost fixed point variable Y that occurs infinitely often in the play is a ν-variable. Otherwise, $\forall$ wins. There may be many different plays; $\exists$ may win some and lose others. A *strategy* for a player is a function which, given a play so far and a position where there is a choice, returns a specific choice and so tells the player how to move. A *history-free* (positional or memoryless) strategy only depends on the current position and not on the previous history of the play: for $\exists$ it is just a choice function. A *winning strategy* is one which, if followed, guarantees that the player will win all plays of the game. Now the fundamental semantic theorems, theorems 3.1 and 3.2, are equivalent to the following.

Theorem 4.6 *$s \models \phi$ iff $\exists$ has a history-free winning strategy for the game $G(s, \phi)$.*

The model checking game on finite structures can be abstracted into a simple two player graph game, called *the parity game*. The state set Q of the graph are the positions and are partitioned into $Q_\forall$ and $Q_\exists$. There is an initial state $q_0 \in Q$. Edges decide possible next positions; for instance, $\exists$ chooses a successor from a vertex $q \in Q_\exists$ and to ensure play is always infinite winning positions have self-loops. The acceptance condition F is just given as a parity condition: F maps each state q of the automaton to a priority $F(q) \in \mathbb{N}$ and $\exists$ wins an infinite play if the least priority that occurs infinitely often is even. The *model-checking* problem for $L\mu$ over finite structures, whether $s_0 \models \phi_0$, is equivalent to solving the parity game (does $\exists$ win q_0 ?). Parity games are determined (i.e. either $\exists$ or $\forall$ has a winning strategy), and a winning strategy is history-free. However, the exact complexity of solving a parity game is a significant open problem.

There is a more intimate connection between $L\mu$ and parity games. An $L\mu$ formula, itself, *is* a parity game as we shall see in section 5; alternating automata are games. Tree automata are games following Gurevich and Harrington [30]. Consider a run of a tree automaton on an n-branching infinite tree whose nodes are labelled with elements of Σ. It starts at the root of the tree with the initial automaton state. If the automaton is in state q examining a node v of the tree labelled with $a \in \Sigma$ then $\exists$ chooses a tuple $(q'_1, \ldots, q'_n)$ that belong to $\delta(q, a)$. Now $\forall$ chooses a direction $i : 1 \leq i \leq n$ so the next position is the ith child of v in state q'_i. The play continues forever. The play is won by $\exists$ if it obeys the acceptance condition. Clearly, $\exists$ has a winning strategy, iff the automaton accepts the tree. (If the acceptance condition is a Rabin condition, this strategy is not history-free; however, it only depends on the 'latest appearance record', an ordering of the automaton states capturing the last time each automaton state occurred in the current play.)

5 Decidability of satisfiability

As with any logic, a key question is decidability of satisfiability, that is, deciding whether a closed formula has a model. A connected property is the *finite model property (fmp)*, that is, if a formula has a model, then it has a finite model. If a logic has the fmp (and the size of the finite model for a formula is effectively bounded), then decidability follows, since one can just check all models up to the size bound. $L\mu$, as we have seen, has the tree model property.

A direct approach to proving decidability of satisfiability is to employ semantic tableaux, to begin with an initial closed formula ϕ in positive normal form and then to build a tree model for it whose states are labelled with locally consistent subsets of the Fisher-Ladner closure of ϕ, $\mathrm{cl}(\phi)$: for instance, if $\psi \wedge \gamma \in s$ then $\psi \in s$ and $\gamma \in s$. Children of a node s are generated using modal successor principles. For each $\langle a \rangle \psi \in s$ create a child node t such that $s \xrightarrow{a} t$ and $\psi \in t$: in turn, $s \xrightarrow{a} t$ when $[a]\psi \in s$ implies $\psi \in t$ and $\psi \in t$ and $\langle a \rangle \psi \in \mathrm{cl}(\phi)$ implies $\langle a \rangle \psi \in s$. This guarantees that the tree has bounded branching degree because $\mathrm{cl}(\phi)$ is finite. Fixed point formulae are "unfolded": ${}^{\mu}_{\nu}X.\,\psi \in s$ implies $\psi({}^{\mu}_{\nu}X.\,\psi) \in s$. The valuation $\mathfrak{V}_{\mathrm{Prop}}$ is then defined: $s \in \mathfrak{V}_{\mathrm{Prop}}(P)$ if and only if $P \in s$.

If ϕ is satisfiable then the construction will generate a finite tree model or an infinite tree that is a pre-model. In the latter case, the problem is how to ensure that it is well-founded. So far, there is no distinction between least and greatest fixed points. As mentioned, an important semantic principle is Park's fixed point induction rule, if $\models \phi(\psi) \Rightarrow \psi$ then $\models \mu X.\phi(X) \Rightarrow \psi$: it follows directly from the semantics because μ is indeed the least pre-fixed point. A question is how to use this semantic principle to guide the tableau construction in such a way that if the starting formula is satisfiable then a model is generable. The following proposition is useful.

Proposition 5.1 *If $\gamma \wedge \mu X.\psi(X)$ is satisfiable and X is not free in γ, then $\gamma \wedge \psi(\mu X.\neg\gamma \wedge \psi)$ is satisfiable.*

Proof. Assume that $\gamma \wedge \mu X.\psi(X)$ is satisfiable but $\models \psi(\mu X.\neg\gamma \wedge \psi) \Rightarrow \neg\gamma$. Therefore, $\models \psi(\mu X.\neg\gamma \wedge \psi) \Rightarrow \neg\gamma \wedge \psi(\mu X.\neg\gamma \wedge \psi)$. Using the fact that $\models \phi'(\mu X.\phi'(X)) \Rightarrow \mu X.\phi'(X)$ and propositional reasoning, $\models \psi(\mu X.\neg\gamma \wedge \psi) \Rightarrow \mu X.\neg\gamma \wedge \psi$. By fixed point induction, $\models \mu X.\psi \Rightarrow \mu X.\neg\gamma \wedge \psi$ and consequently $\models \mu X.\psi \Rightarrow \neg\gamma$ which is a contradiction.

5.1 The aconjunctive fragment

The tableau approach was employed by Kozen [35] to decide satisfiability. Unfortunately, he could only prove the result for a sublogic of $L\mu$, when the starting formula ϕ is *aconjunctive*: that is, if $\mu X.\psi$ is a subformula of ϕ and $\psi_1 \wedge \psi_2 \in \mathrm{cl}(\mu X.\psi)$ then for at most one ψ_i is it the case that $\mu X.\psi \in \mathrm{cl}(\psi_i)$. For instance, $\nu Z.\mu X.([a]X \vee \langle b\rangle Z) \wedge \langle a\rangle Z$ is aconjunctive: the subformula $\gamma = \mu X.([a]X \vee \langle b\rangle Z) \wedge \langle a\rangle Z$ has one conjunction in its closure $([a]\gamma \vee \langle b\rangle Z) \wedge \langle a\rangle Z$ and γ is only in the closure of the first conjunct. In contrast, $\gamma = \mu X.\nu Z.([a]X \vee \langle b\rangle Z) \wedge \langle a\rangle Z$ fails to be aconjunctive: γ is in the closure of both conjuncts $([a]\gamma \vee \langle b\rangle(\nu Z.[a]\gamma \wedge \langle a\rangle Z)) \wedge \langle a\rangle(\nu Z.[a]\gamma \wedge \langle a\rangle Z)$. Aconjunctivity restricts how a formula $\mu X.\psi \in \mathrm{cl}(\phi)$ can regenerate itself in the tableau construction: there can only be a linear pattern of regeneration (as opposed to the more general branching pattern for full $L\mu$). In the case of $\gamma = \nu Z.\mu X.([a]X \vee \langle b\rangle Z) \wedge \langle a\rangle Z$, the relevant formula $\gamma' = \mu X.([a]X \vee \langle b\rangle\gamma) \wedge \langle a\rangle\gamma$ can only regenerate itself through the the subformula $[a]X$: so, multiple regenerations of γ' happen only as part of a linear sequence. On the other hand, $\gamma = \mu X.\nu Z.([a]X \vee \langle b\rangle Z) \wedge \langle a\rangle Z$ can regenerate itself through both subformulae $([a]X \vee \langle b\rangle Z)$ and $\langle a\rangle Z$: so, multiple regenerations of γ form a tree.

Given the aconjunctive restriction, one can guide the construction of the tree model by applying proposition 5.1 to $\mu X.\psi \in s$: as it is unfolded its interpretation is strengthened to $\psi(\mu X.\neg s \wedge \psi)$ where s abbreviates the conjunction of all formulas in s. The strengthening interpretation is extended as $\mu X.\psi$ regenerates itself in descendent states t of s, so that an unfolding in t is re-interpreted as $\psi(\mu X.\neg s \wedge \ldots \wedge \neg t \wedge \psi)$ thereby ensuring that a descendent state within which $\mu X.\psi$ is regenerated cannot have the same labelling as the ascendent state (and because the starting formula is aconjunctive this will guarantee a well-founded pre-model). To do this, one needs a careful ordering on fixed point subformulae (in terms of which are more outermost) to ensure that the set of labellings remains finite. Kozen showed that the decision procedure for this fragment (that contains PDL and CTL) works in exponential time and at the same time the proof delivers the finite model property. In fact, the construction works for a more generous fragment of the logic, called the *weak* aconjunctive fragment in [71]. One only needs to guarantee that there is a linear pattern of regeneration of least fixed point subformulae relative to each individual branch in the tree model. The formula $\gamma = \mu X.\nu Z.\langle a\rangle X \wedge \langle a\rangle Z$ belongs to this more generous fragment because the regenerations of γ through the subformulae $\langle a\rangle X$ and $\langle a\rangle Z$ happen in different branches: the formula $\mu X.\nu Z.([a]X \vee \langle b\rangle Z) \wedge \langle a\rangle Z$ does not belong to it. In fact, every closed formula of $L\mu$ is semantically equivalent to a weak aconjunctive formula (which follows from results below). However, it is an open question whether the tableau technique can be made to work directly for all formulae.

5.2 Towards automata

The first decision procedure for full $L\mu$ reduced the problem to SnS, the second-order theory of n-successors, [38]. The structure for SnS is the transition system (tree) with state space $\{0, \ldots, n-1\}^*$ and transition relations $w \xrightarrow{i} wi$ for each $i < n$. Büchi showed that the monadic second-order (MSO) theory of S1S is decidable [13]: besides first-order constructs, MSO has quantifiers over sets of states. S1S is a weak form of arithmetic where, in this presentation, the number n is 0^n and $\xrightarrow{0}$ is the $+1$ function. Rabin extended Büchi's result by showing that the MSO theory of SnS is decidable for any $n > 0$ [60]. Kozen and Parikh's proof of decidability of satisfiability for full $L\mu$ uses the tree model property with bounded branching degree. Given a formula ϕ the maximum required branching degree n can be calculated from $\mathrm{cl}(\phi)$. The formula ϕ can then be translated almost directly into SnS by examining its semantics: for instance, $\|\nu X.\phi\|_{\mathfrak{V}} = \exists S.S \subseteq \|\phi\|_{\mathfrak{V}[X:=S]}$ and $\|X\|_{\mathfrak{V}} = \mathfrak{V}(X)$. Elements of Prop are treated as variables (and are existentially quantified over). Labels in diamond modalities are judiciously mapped to "directions" $i < n$ and labels in box modalities to sets of directions. For example, $\nu X.\langle a\rangle(X \wedge \neg P) \wedge \langle a\rangle(X \wedge P)$ is translatable into the S2S formula

$$\exists P.\exists S.\forall x.\exists y.\exists z.(x \in S \Rightarrow x \xrightarrow{1} y \wedge y \in S \wedge y \notin P) \wedge (x \in S \Rightarrow x \xrightarrow{2} z \wedge z \in S \wedge z \in P)$$

The formula ϕ is satisfiable if and only if its translation is true in SnS: for instance, the S2S formula above is true. The key feature in the MSO decidability proofs is that in a formula $\exists X.\phi$, quantification can be restricted to "regular" sets of states which leads to quantifier elimination when the normal form is a nondeterministic finite state automaton. In the case of S1S it is a Büchi word automaton and in the more general setting of SnS it is a Rabin tree automaton: these automata are defined in section 4.3. The automaton normal form for $\forall X.\phi$, that is $\neg\exists X \neg\phi$, involves an exponential increase in size because of complementation. Consequently, the decision procedure for SnS, $n > 0$, is (and must be) non-elementary. Because $\mu X_1.\nu X_2.\ldots\mu X_m.\nu X_{m+1}.\psi$ is translated into the MSO formula $\forall S_1.\exists S_2\ldots\forall S_m.\exists S_{m+1}.\psi'$, Kozen and Parikh's decision procedure for $L\mu$ is also non-elementary.

MSO formulae with second-order quantification, unlike fixed point formulae, are expressively succinct. A direct translation of $L\mu$ formulae into finite state automata, without intervening MSO formulae, could lead to a more efficient decision procedure. With this technique Streett provided an elementary time decision procedure for PDL with looping and converse [65]. With Emerson he employed the same technique for $L\mu$ and obtained a decision procedure for satisfiability and a proof of the finite model property at once [64]. The procedure is in elementary time. The central ingredient (besides the tree model property) is the relationship between $L\mu$ and Rabin automata, which is established using the fundamental semantic theorem. For, the constraint on fixpoint regeneration and infinite repetition is expressible as a Rabin acceptance condition. Now we can construct an automaton that accepts such bounded-branching tree models, by combining a finite-state automaton to check the local consistency (that is, to check that the putative model is a pre-model), and a Rabin automaton to check that the pre-model is well-founded. Thus the formula is satisfiable if this product automaton accepts some tree. Now automata theory, see for instance [66], tells us that (a) this question is decidable (b) if such an automaton accepts some tree, it accepts a regular tree, that is, one that is the unravelling of a finite system; this gives the results. Later, Emerson and Jutla provided an exponential time decision procedure (which is optimal) using an improved determinization construction and an improved tree automata emptiness test [22]: there is an exponential (in the size of the formula) bound on the size of the model.

5.3 Alternating parity automata

There is a slight mismatch between $L\mu$ models and SnS models because of the fixed branching degree and the explicit indexed successors. However, it is possible to define automata that can directly recognise $L\mu$ models by navigating through their transition graphs. We define *alternating parity automata* for this purpose following, for example, [40]. The only restriction is that we assume that Prop is a finite set (those propositions that appear in a starting formula ϕ). A rooted model for a closed formula ϕ is a triple $(\mathfrak{T}, s_0, \mathfrak{V})$.

Recall the notion of automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ as defined in section 4.3. Now think of the transition function of an automaton as a logical formula. For a word automaton, if $\delta(q, a) = \{q_1, \ldots, q_m\}$ then it is the formula $q_1 \vee \ldots \vee q_n$. For a tree automaton if $\delta(q, a) = \{(q_1^1, \ldots, q_n^1), \ldots, (q_1^m, \ldots, q_n^m)\}$ then it is $((1, q_1^1) \wedge \ldots \wedge (n, q_n^1)) \vee \ldots \vee ((1, q_1^m) \wedge \ldots \wedge (n, q_n^m))$: here the element (i, q') means create an ith-child with label q'. A word or tree is accepted if there exists an accepting run for that word or tree; hence, the disjuncts. However, for a tree, every path through it must be accepting; hence the conjuncts. In *alternating* word automata, the transition function is given as an arbitrary boolean expression over states: for instance, $\delta(q, a) = q_1 \wedge (q_2 \vee q_3)$. In alternating tree automata it is a boolean expression over directions and states: for instance, $((1, q_1) \wedge (1, q_2)) \vee (2, q_3)$. Now the definition of a run becomes a tree in which, successor transitions obey the boolean formula. In particular, even for an alternating automaton on words, a run is a tree, and not just a word. The acceptance criterion is as before, that every path of the run must be accepting. An alternating automaton is just a game too where $\forall$ is responsible for $\wedge$ choices and $\exists$ for $\vee$ choices (as in section 4.4).

The idea now is to replace pairs (i, q') with simple modal formulae. We define *modal* automata whose transition functions appeal to a modal language (similar to modal equation systems). Formally, a modal automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is the set $\wp\mathrm{Prop}$ and F is the parity acceptance condition. The transition function $\delta : Q \times \Sigma \rightarrow B$ where B is the following set of positive modal formulae (with modal depth at most 1).

- $\top, \bot$ are in B
- If $q \in Q$ and $a \in \mathcal{L}$ then $\langle a\rangle q$ and $[a]q$ are in B
- If B_1 and B_2 are in B then $B_1 \vee B_2$ and $B_1 \wedge B_2$ are in B

The automaton traverses the modal model, starting at s_0 and moving from a state s to successor states t such that $s \xrightarrow{a} t$ for some $a \in \mathcal{L}$, according to the transition function. However, not every successor may be included and some successors may be included multiple times: for instance, if q is the current automaton state for s and $\delta(q, \mathrm{Prop}(s)) = \langle a\rangle p_1 \wedge [c]p_2$ [3] and $s \xrightarrow{a} s_1$, $s \xrightarrow{b} s_2$, $s \xrightarrow{c} s_1$ and $s \xrightarrow{c} s$ then the automaton changes state to p_1 and moves to s_1 in the model, changes to p_2 and also moves to s_1 in the model and changes to p_2 and remains at s. As with tree automata, a run of $\mathcal{A}$ on a model is a a labelled tree $(N, \xrightarrow{i}, L')$ where $N \subseteq \omega^*$ that obeys the tree property that if $wi \in N$ then $w \in N$ and $w \xrightarrow{i} wi$: a node $x \in N$ may have infinitely many successors $xi \in N$, as models have no bound on their branching degree. Unlike tree automata, there is no requirement that complete branches should have infinite length.

In more detail, a run of $\mathcal{A}$ on a modal model is a projection of an intermediate structure, a tree with composite labels $(N, \xrightarrow{i}, L)$. The labelling $L : N \rightarrow S \times Q$ where S is the state space of the model: for the root of the tree, $L(\epsilon) = (s_0, q_0)$. The labels of a node and its successors have to obey the transition function. First, given a state s of the model let M_s range over mixture subsets $\{(t, q') \mid s \xrightarrow{a} t \text{ for some } a \text{ and } q' \in Q\}$. Next, we define when a subset M_s satisfies a modal formula B, which we write $M_s \models B$.

$$\begin{array}{ll}
M_s \models \top & M_s \not\models \bot \\
M_s \models \langle a\rangle p & \text{iff } \exists t.s \xrightarrow{a} t \text{ and } (t,p) \in M_s \\
M_s \models [a]p & \text{iff } \forall t. \text{ if } s \xrightarrow{a} t \text{ then } (t,p) \in M_s \\
M_s \models B_1 \vee B_2 & \text{iff } M_s \models B_1 \text{ or } M_s \models B_2 \\
M_s \models B_1 \wedge B_2 & \text{iff } M_s \models B_1 \text{ and } M_s \models B_2
\end{array}$$

Given $\mathcal{A}$ and a rooted model, one grows a labelled tree from the root ϵ with $L(\epsilon) = (s_0, q_0)$. If $L(x) = (s, q)$ and $\delta(q, \mathrm{Prop}(s)) = B$ then there is a (possibly empty) set M_s such that $M_s \models B$. A child of x is produced for each element of M_s: that is, $M_s = \{L(xi) \mid xi \in N\}$. For example, if $L(x) = (s, q)$ and $\delta(q, \mathrm{Prop}(s)) = \langle a\rangle q_1 \wedge [a]q_2 \wedge \langle a\rangle q_3$ and in the model $s \xrightarrow{a} s_i$ for all $i \geq 0$ then a candidate M_s is $\{(s_0, q_1), (s_1, q_3), (s_0, q_2), \ldots, (s_i, q_2), \ldots\}$: here there are infinitely many such candidates. The required run is the projection of the tree to states in Q, the tree $(N, \xrightarrow{i}, L')$ whose labelling $L'(x) = q$ if $L(x) = (s, q)$ for some s. A run is *accepting* if all (labellings of) infinite branches starting from the root obey the parity acceptance condition F.

Given a rooted model $(\mathfrak{T}, s_0, \mathfrak{V})$, $s_0 \models \mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on that model. The following is relatively straightforward (and is reminiscent of translating to and from boolean equation systems).

Theorem 5.2 *For each modal automaton there is an equivalent closed formula of $L\mu$, and for each closed*

[3] $\mathrm{Prop}(s)$ is the subset of Prop true at s.

formula of $L\mu$ there is an equivalent modal automaton.

5.4 Automaton normal form

We can now extract a semantic *normal form* for $L\mu$ due to Walukiewicz [71,32]. If Γ is a finite set of formulae, $(a)\Gamma$ abbreviates $\bigwedge_{\phi\in\Gamma}\langle a\rangle\phi \wedge [a]\bigvee_{\phi\in\Gamma}\phi$. Every modal automaton is equivalent to a *restricted* modal automaton. Let $\Sigma = \{a_1,\ldots,a_n\}$. The transition function is restricted: formulae of B are disjunctions of conjuncts of the form $(a_1)B_1 \wedge \ldots \wedge (a_n)B_n$ where each $B_i \subseteq Q \cup \{\top\}$. The proof of the following is far from trivial and depends on the combinatorial features of automata, especially determinization.

Theorem 5.3 *For each modal automaton there is an equivalent restricted modal automaton.*

A formula is in *automaton normal form (anf)* [4] , if it belongs to the following sublogic, where

- P, $\neg P$ and Z are anfs
- If ϕ_1 and ϕ_2 are anfs, so is $\phi_1 \vee \phi_2$
- If ϕ is an anf, then so are $\nu Z.\phi$, $\mu Z.\phi$
- If each Γ_i is a finite set of anfs and $a_i \neq a_j$ when $i \neq j$ and α^+ is a finite set of atomic propositions and their negations, then $(a_1)\Gamma_1 \wedge \ldots \wedge (a_n)\Gamma_n \wedge \alpha^+$ is an anf

Anf formulae are the characteristic formulae for restricted automata. For instance, a clause $\{\langle a\rangle p, \langle a\rangle q\}$ with respect to labels a, b becomes the formula $\langle a\rangle p \wedge \langle a\rangle q \wedge [a](p \vee q) \wedge [b]\bot$.

Proposition 5.4 *For each restricted automaton there is an equivalent anf formula.*

Therefore, anfs are semantic normal forms for $L\mu$. We can effectively construct the anf normal form for a formula ϕ in positive normal form. First, use Theorem 5.2 to build an equivalent modal automaton $\mathcal{A}_\phi$ for ϕ. Next, use Theorem 5.3 to transform $\mathcal{A}_\phi$ into an equivalent restricted automaton $\mathcal{A}^+_\phi$. Finally, use Proposition 5.4 to convert $\mathcal{A}^+_\phi$ into an equivalent anf formula ϕ'.

An anf formula is weakly aconjunctive (although not necessarily aconjunctive). After simplification, the anf normal form of the earlier formula $\mu X.\nu Z.([a]X \vee \langle b\rangle Z) \wedge \langle a\rangle Z$ that is not aconjunctive is $\mu X.\nu Z.(a)\{X\} \vee ((a)\{Z,\top\} \wedge (b)\{Z,\top\})$. In fact, conjunction is even more constraining in anf formulae. Consider, the semantic tableau construction for an anf formula ϕ. The only time we need to apply $\wedge$ decomposition is just before the application of modal successors: assume a state s is labelled with the formula $(a_1)\Gamma_1 \wedge \ldots \wedge (a_n)\Gamma_n \wedge \alpha^+ \in \mathrm{cl}(\phi)$. At s it reduces to its components $(a_1)\Gamma_1,\ldots,(a_n)\Gamma_n,\alpha^+$. If α^+ is consistent then modal children $s \xrightarrow{a_i} t$ are created: however, by the definition of (a) each modal successor t is labelled with a single anf formula in $\mathrm{cl}(\phi)$. Therefore, as shown by Walukiewicz, ϕ is satisfiable iff all its fixed point subformulae $\mu X.\psi(X)$ are replaced with $\psi(\bot)$ and all subformulae $\nu X.\psi(X)$ are replaced with $\psi(\top)$. To illustrate this, assume $\phi = \mu X.\psi$ is satisfiable. Consider a rooted model and a least ordinal o such that $s_0 \models \mu X^o.\psi$. Consider its semantic tableau with initial state s_0 labelled with ϕ. If there is a descendent state t that is also labelled with ϕ then $t \models \mu X^{o'}.\psi$ with $o' < o$ which contradicts that o is least. Therefore, there is a model for ϕ such that no descendent state is labelled with ϕ, which is, therefore, also a model for $\psi(\bot)$. Consequently, satisfiability checking for an anf formula can be done in linear time [32]. To obtain the fmp for anf ϕ, replace each subformulae $\mu X.\psi(X)$ with $\psi(\bot)$ and build a semantic tableaux for it. For modal successors, if at state s, $\nu X.\psi \in \Gamma$ and $(a)\Gamma \in s$ and some state t is on the path from the root to s and t is labelled with $\nu X.\psi$ then let $s \xrightarrow{a} t$: in this way, a finite model for ϕ is constructed.

[4] Walukiewicz terms them “disjunctive formulae”.

6 Complete axiomatization

A related problem to decidability is the question of providing an axiomatization of the theory of the modal mu-calculus. In his original paper, Kozen presented the axiomatization as an equational theory which is equivalent to the following.

(i) axioms and rules for minimal multi-modal logic K

(ii) $\phi(\mu X.\phi(X)) \Rightarrow \mu X.\phi(X)$

(iii) $\dfrac{\phi(\psi) \Rightarrow \psi}{\mu X.\phi(X) \Rightarrow \psi}$

Axiom ii is the axiom for a least fixed point that its "unfolding" implies it and rule iii is Park's fixed point induction rule. The duals of this axiom and rule for greatest fixed points are; $\nu X.\phi(X) \Rightarrow \phi(\mu X.\phi(X))$ and if $\psi \Rightarrow \phi(\psi)$ then $\psi \Rightarrow \nu X.\phi(X)$.

However, despite the naturalness of this axiomatization, Kozen was unable to show that it was complete. He was, however, able to show completeness for the aconjunctive fragment. In fact, a proof works for *weak* aconjunctive formulae using the consistency version of proposition 5.1: if $\gamma \wedge \mu X.\psi(X)$ is consistent [5] and X is not free in γ, then $\gamma \wedge \psi(\mu X.\neg\gamma \wedge \psi)$ is consistent. The proof is similar to the tableau construction described in section 5.1. Given an initial consistent formula in positive normal form one builds a tree model: the construction is guided by the proposition above as in the satisfiability proof.

Completeness for the full language remained open for more than a decade, until it was finally solved by Walukiewicz in [71], who established that Kozen's axiomatization is indeed complete. The proof is very involved and, in effect, internalises the automata theoretic satisfiability proof described earlier. It utilises automata normal form and weak aconjunctivity. It is more straightforward (as with satisifiability) to show using tableaux that if an anf formula is consistent then it has a model. Much harder to prove is that every (closed) formula is provably equivalent within the axiom system to an anf formula. Walukiewicz utilises games on infinite tableaux to show this.

The following are valid fixpoint principles (which, by duality also are true for ν).

$$\mu X.\mu Y.\phi(X,Y) \iff \mu X.\phi(X,X) \iff \mu Y.\mu X.\phi(X,Y)$$

Arnold and Niwinski call these "the golden lemma" of μ-calculus [5]. Other interesting valid fixpoint principles include $\mu X.\phi(X) \Rightarrow \nu X.\phi(X)$, by monotonicity, and the following, due to Niwinski, that generalises that 'almost always' implies 'infinitely often'.

$$\mu X.\nu Y.\phi(X,Y) \Rightarrow \nu Y.\mu X.\phi(X,Y)$$

Deriving these principles deductively from Kozen's complete axiom system is by no means easy (as opposed to their derivations using the semantics).

7 Alternation

As we said earlier, the alternation of fixpoints is what gives $L\mu$ its expressive power, and also what appears to generate the computation complexity of model-checking. In this section, we study alternation in more detail. As we have said, the idea is to count alternations of minimal and maximal fixpoint operators, but to do so in a way that only counts real dependency. The paradigm is 'always eventually' versus 'infinitely often': the 'always eventually' formula

$$\nu Y.(\mu Z.P \vee \langle a\rangle Z) \wedge \langle a\rangle Y$$

[5] A formula ϕ is consistent with respect to an axiom system if $\phi \Rightarrow \bot$ is not derivable within the axiom system. Completeness of an axiom system is equivalent to the statement that every consistent formula has a model (is satisfiable).

is, using a straightforward model-checking algorithm, really no worse to compute than two disjoint fixpoints, since the inner fixpoint can be computed once and for all, rather than separately on each outer approximant; on the other hand, the 'infinitely often' formula

$$\nu Y.\mu Z.(P \vee \langle a\rangle Z) \wedge \langle a\rangle Y$$

really does need the full double induction on approximants.

The definition of Emerson and Lei takes care of this by observing that the 'eventually' subformula is a closed subformula, and giving a definition that ignores closed subformulae when counting alternations. The stronger notion of Niwiński, which also has the advantage of being robust under translation to modal equation systems, also observes that, for example, $\mu X.\nu Y.[-]Y \wedge \mu Z.[-](X \vee Z)$ although it looks like a $\mu/\nu/\mu$ formula, is morally a μ/ν formula, since the inner fixpoint does not refer to the middle fixpoint.

The alternation depth referred to in the complexity of model-checking is a measure of alternation that is symmetric in μ and ν. It is possible to give algorithms that compute the alternation depth of a formula [24,1,34], and this is how the notion was presented by Emerson and Lei. However, for our purposes it is easier to start from a definition of classes for formula, formalizing the idea of 'a $\mu/\nu/\mu$ formula' etc.; such a definition is analogous to the usual definition of quantifer alternation for predicate logic, an analogy which will be exploited later. This was how Niwiński [53] presented the notion of alternation, and we follow his presentation.

Assuming positive form, a formula ϕ is said to be in the classes $\Sigma_0^{N\mu}$ and $\Pi_0^{N\mu}$ iff it contains no fixpoint operators. To form the class $\Sigma_{n+1}^{N\mu}$ (resp. $\Pi_{n+1}^{N\mu}$), take $\Sigma_n^{N\mu} \cup \Pi_n^{N\mu}$, and close under the following rules:

(i) if $\phi_1, \phi_2 \in \Sigma_{n+1}^{N\mu}$ (resp. $\Pi_{n+1}^{N\mu}$), then $\phi_1 \vee \phi_2, \phi_1 \wedge \phi_2, \langle a\rangle\phi_1, [a]\phi_1 \in \Sigma_{n+1}^{N\mu}$ (resp. $\Pi_{n+1}^{N\mu}$);

(ii) if $\phi \in \Sigma_{n+1}^{N\mu}$ (resp. $\Pi_{n+1}^{N\mu}$), then $\mu Z.\phi \in \Sigma_{n+1}^{N\mu}$ (resp. $\nu Z.\phi \in \Pi_{n+1}^{N\mu}$);

(iii) if $\phi(Z), \psi \in \Sigma_{n+1}^{N\mu}$ (resp. $\Pi_{n+1}^{N\mu}$), then $\phi(\psi) \in \Sigma_{n+1}^{N\mu}$ (resp. $\Pi_{n+1}^{N\mu}$), *provided* that no free variable of ψ is captured by a fixpoint operator in ϕ.

If we omit the last clause, we get the definition of 'simple-minded' alternation $\Sigma_n^{S\mu}$, that just counts syntactic alternation; if we modify the last clause to read '... *provided* that ψ is a closed formula', we obtain the Emerson–Lei notion $\Sigma_n^{EL\mu}$. (We write just Σ_n^{μ} when the distinctions are not important, or when we are making a statement that applies to all versions.)

To get the symmetrical notion of *alternation depth* of ϕ, we can define it to be the least n such that $\phi \in \Sigma_{n+1}^{\mu} \cap \Pi_{n+1}^{\mu}$. To make these definitions clear, consider the following examples:

- The 'always eventually' formula is $\Pi_2^{S\mu}$, but not $\Sigma_2^{S\mu}$, and so its simple alternation depth is 2. However, in the Emerson–Lei notion, it is also $\Sigma_2^{EL\mu}$, since $\nu Y.W \wedge \langle a\rangle Y$ is $\Pi_1^{EL\mu}$ and so $\Sigma_2^{EL\mu}$, and by substituting the closed $\Sigma_2^{EL\mu}$ (and in fact $\Sigma_1^{EL\mu}$) formula $\mu Z.P \vee \langle a\rangle Z$ for W we get 'always eventually' in $\Sigma_2^{EL\mu}$; hence its Emerson–Lei (and Niwiński) alternation depth is 1.
- The 'infinitely often' formula is Σ_2^{μ} but not Π_2^{μ}, in all three definitions, and so has alternation depth 2.
- The formula $\mu X.\nu Y.[-]Y \wedge \mu Z.[-](X \vee Z)$ is $\Sigma_3^{S\mu}$, but not $\Pi_3^{S\mu}$; it is also $\Sigma_3^{EL\mu}$ but not $\Pi_3^{EL\mu}$, since there are no closed subformulae to bring the substitution clause into play. However, in the Niwiński definition, it is actually $\Sigma_2^{N\mu}$: $\nu Y.[-]Y \wedge W$ is $\Pi_1^{N\mu}$ and so $\Sigma_2^{N\mu}$; we can substitute the $\Sigma_1^{N\mu}$ formula $\mu Z.[-](X \vee Z)$ for W without variable capture, and so $\nu Y.[-]Y \wedge \mu Z.[-](X \vee Z)$ is $\Sigma_2^{N\mu}$; and now we can add the outer fixpoint, still remaining in $\Sigma_2^{N\mu}$.

A natural question is whether the hierarchy of properties definable by Σ_n^{μ} formulae is actually a strict hierarchy, or whether the hierarchy collapses at some point so that no further alternation is needed. This problem remained open for a while; by 1990, it was known that $\Sigma_2^{N\mu} \neq \Pi_2^{N\mu}$ [4]. No further advance was made until 1996, when the strictness of the hierarchy was established by Bradfield [8,9,11].

Theorem 7.1 *For every n, there is a formula $\phi \in \Sigma_n^\mu$ which is not equivalent to any Π_n^μ formula.*

Bradfield established this for $\Sigma_n^{N\mu}$, which implies the result for the other two notions. At the same time, Lenzi [43] independently established a slightly weaker hierarchy theorem for $\Sigma_n^{EL\mu}$.

Lenzi's proof is technically complex, and the underlying stratagem is not easy. Bradfield's proof appears technically complex, but most of the complexity is really just routine recursion-theoretic coding; the underlying stratagem is quite simple, and in some ways surprising. If one takes first-order arithmetic, one can add fixpoint operators to it, and one can then define a fixpoint alternation hierarchy in arithmetic. A standard coding and diagonalization argument shows that this hierarchy is strict [9]. The trick now is to transfer this hierarchy to $L\mu$. Simply by writing down the semantics, it is clear (give or take some work to deal with the more complex notions of alternation) that if one takes a recursively presented transition system and codes it into the integers, then for a modal formula $\phi \in \Sigma_n^\mu$, its denotation $\|\phi\|$ is describable by an arithmetic Σ_n^μ formula. However, it is also possible, given any arithmetic fixpoint formula χ, to build a transition system and a modal formula ϕ, of the same alternation depth as χ, such that $\|\phi\|$ is characterized by χ. If we take χ to be a strict Σ_n^μ arithmetic formula, then no Π_n^μ arithmetic formula is equivalent to it, and therefore no Π_n^μ modal formula can be equivalent to ϕ. The transition system that is constructed is infinite, but by the finite model property, the hierarchy transfers down to the class of finite models.

Both proof techniques construct explicit examples of hard formulae. Bradfield's examples have the form

$$\mu X_n.\nu X_{n-1}.\ldots\mu X_1.[c]X_1 \vee \langle a_1\rangle X_1 \vee \ldots \vee \langle a_n\rangle X_n.$$

There are further questions one can ask about the alternation hierarchy. For example, is it still strict even on the binary tree? The answer is yes, given independently by Bradfield [10,11] and Arnold [3] – the latter also gives a nice alternative proof of the main theorem, using topological methods rather than computability methods.

A more interesting question, and one that is still open, is: given a formula, what is its 'semantic' alternation depth? That is, what is the least alternation depth of any equivalent formula? Otto [55] showed that it is decidable whether a formula is equivalent to an alternation-free formula, and then Küsters and Wilke showed [41] it for alternation depth 1. Decidability is not known for higher levels.

8 Bisimulation invariance

A hallmark of modal logic is bisimulation invariance: if $s \models \phi$ and s and s' are bisimulation equivalent then $s' \models \phi$. As we have seen, this remains true for $L\mu$ formulae. In logic, in general, structures are viewed as equivalent when they are isomorphic. However, in computation when structures represent behaviour of systems weaker forms of equivalence, such as automata acceptance equivalence or bisimulation equivalence, are more appropriate; see, for example, Milner [51].

8.1 $L\mu$ and MSOL

A modal formula can be translated into an equivalent bisimulation invariant first-order logic formula (over transition graphs) with one free variable. The translation is merely the semantics. Let $\phi[x]$ be the translation of ϕ with free variable x: for instance, $P[x] = P(x)$ and $\langle a\rangle\phi[x] = \exists y.\, x \xrightarrow{a} y \wedge \phi[y]$. Clearly, $s \models \phi$ iff $\phi[s]$ holds. Van Benthem proved the converse: a bisimulation invariant first-order logic formula with one free variable is equivalent to a modal formula. Modal logic is the bisimulation invariant fragment of first-order logic.

The question is whether there is a similar result for closed formulae of $L\mu$. As we have seen, there is an intimate relationship between $L\mu$ and automata, games or SnS. None of these notations provide an obvious semantics for $L\mu$ formulae. Monadic second-order logic (MSOL) of transition graphs extends first-order logic with quantification over monadic predicates. With this addition we can translate $L\mu$.

$$\nu Z.\phi[x] \;=\; \exists Z.(\forall y.Z(y) \Rightarrow \phi[y]) \Rightarrow Z(x)$$

So, an $L\mu$ formula is translated into an equivalent bisimulation invariant MSOL formula with one free variable. Remarkably, the converse is also true, as proved by Janin and Walukiewicz [33].

Theorem 8.1 *A bisimulation invariant MSOL formula with one free variable is equivalent to an $L\mu$ formula.*

In other words, $L\mu$ is the bisimulation invariant fragment of MSOL.

The proof of this theorem is intricate and again illustrates the potency of automata. The authors define an *ω-expansion* of a rooted model which is like the usual unravelling of the system into a tree, with the addition that the tree contains ω-many copies of every successor node. If $\phi(x)$ is a bisimulation invariant MSOL formula and $\phi(s)$ holds where s is the root of a model then $\phi(s)$ remains true for the ω-expanded model.

The proof uses modal automata from section 5.3. The transition function is defined using a simple modal language. If the automaton is in state q and at state s in the modal model and $\delta(q, \mathrm{Prop}(s)) = B$ then there is a mixture set $M_s \models B$ where $M_s \subseteq \{(t, q') \mid s \xrightarrow{a} t$ for some a and $q' \in Q\}$. Instead of simple modal formulae B, the automaton could employ first-order formulae with one free variable $B[x]$. Now, for instance, $M_s \models \exists y.x \xrightarrow{a} y \wedge p[y]$ iff $(t, p) \in M_s$ for some t such that $s \xrightarrow{a} t$. Critically, there is also a similar automata characterisation of MSOL formulae on trees. The transition function $\delta : Q \times \Sigma \rightarrow B'$ where B' is very similar to $B[x]$ except that it involves inequalites. When in CNF, formulae $B'[x]$ have the form

$$\exists y_1, \ldots y_n.(\bigwedge_{i\neq j} y_i \neq y_j \wedge x \xrightarrow{a_1} y_1 \wedge p_1[y_1] \wedge \ldots \wedge x \xrightarrow{a_n} y_n \wedge p_n[y_n] \wedge \forall z. \bigwedge z \neq y_i \wedge \psi(z, x)$$

where $\psi(z, x)$ captures the "box" formulae. The inequalities are effectively redundant in an ω-expanded model. The formulae $B'[x]$ collapse to $B[x]$ with respect to these models.

Van Benthem's theorem also holds for finite models: modal logic is the bisimulation invariant fragment of first-order logic with respect to finite models. It is an open question if this is true for $L\mu$ and MSOL.

8.2 Multi-dimensional $L\mu$ and Ptime

A major interest is classifying logics according to their expressive power. Computationally, we can ask whether there are logics that characterize complexity classes. A classic result is that existential second-order logic exactly captures NP properties of finite structures. A key open problem is whether there is such a logic for PTIME properties. (For finite structures *with a linear ordering* the PTIME properties are exactly captured by least fixed point logic of section 9.2.) However, Otto shows that *bisimulation invariant* monadic PTIME properties (of modal structures) is logically characterizable by a multi-dimensional $L\mu$ [54].

For simplicity, assume finite $L\mu$ rooted structures whose label set is a singleton and let Prop be finite. Formulas of $L\mu$ are interpreted with respect to a single state. Consider instead k-tuples of states $(s_1, \ldots, s_k)$. Given such tuples we can define transition relations $\xrightarrow{i}$, for each $i : 1 \leq i \leq k$: $(s_1, \ldots, s_k) \xrightarrow{i} (t_1, \ldots, t_k)$ if $s_i \longrightarrow t_i$ and $s_j = t_j$ for all $j \neq i$. Otto defines the logic $L\mu^k$ (with $L\mu = L\mu^1$). Formulae may contain variables x_i, $1 \leq i \leq k$. Atomic formulae have the form Px_i: $(s_1, \ldots, s_k) \models Px_i$ iff $P \in \mathrm{Prop}(s_i)$. Modal formulae have the form $\langle i\rangle\phi$ and $[i]\phi$. Formulae are closed under boolean connectives. There is a substitution operation $\sigma : \{1, \ldots, k\} \rightarrow \{1, \ldots, k\}$: $\phi\sigma$ is the formula $\phi\{x_{\sigma(1)}/x_1, \ldots, x_{\sigma(k)}/x_k\}$. Finally, fixed points are k-ary: $\mu X(x_1, \ldots, x_k).\phi$ (and are interpreted as in section 9.2). Formulae of $L\mu^k$ are bisimulation invariant. The logic that characterizes bisimulation invariant monadic PTIME are the monadic formulae of $\bigcup_{k>0} L\mu^k$. Crucially, for $k > 1$, bisimulation equivalence is definable in $L\mu^k$.

$$\nu X(x_1, \ldots, x_k). \bigwedge_{P\in\mathrm{Prop}} Px_1 \Leftrightarrow Px_2 \wedge [1]\langle 2\rangle X(x_1, \ldots, x_k) \wedge [2]\langle 1\rangle X(x_1, \ldots, x_k)$$

For canonical finite rooted models (rooted models quotiented with respect to bisimulation equivalence) one can define a linear ordering on states via bisimulation *in*equivalence. So, each PTIME property is definable in least fixed point logic and, in fact, in some $L\mu^k$.

8.3 Bisimulation quantifiers and interpolation

In previous sections, a number of standard logical questions about the $L\mu$ have been covered, such as satisfiability, completeness, etc. These were all addressed, if not solved, early in the history of the logic. There are other standard questions about logics which, perhaps surprisingly, were not addressed until quite recently. In this subsection, we describe briefly work on interpolation theorems and related issues. A key ingredient in these proofs is again alternating parity automata; another ingredient is an interesting notion of 'bisimulation quantifier'.

A logic enjoys the *Craig interpolation property* if whenever $\phi \Rightarrow \psi$, then there is a third formula χ, using only those atomic symbols occurring in both ϕ and ψ, such that $\phi \Rightarrow \chi \Rightarrow \psi$. The *uniform interpolation property* requires further that to find χ, it suffices to know only one of ϕ or ψ and what the common language is. (That is, one can construct the strongest formula implied by ϕ in a given language, or the weakest formula implying ψ in a given language.) Maksimova showed [47] that most common temporal logics do not have interpolation. In [16], d'Agostino and Hollenberg show that $L\mu$ has interpolation, and even uniform interpolation, as we now sketch.

Let ϕ be a sentence, and P an atomic proposition occurring in ϕ. The aim is to construct a formula $\widetilde{\exists} P.\phi$ which is the strongest implicate of ϕ in the language omitting P. This can be done by using results of the Janin–Walukiewicz paper discussed earlier: translate ϕ into an MSOL sentence $\tilde{\phi}$, quantify it (in MSOL) to form $\exists P.\tilde{\phi}$, and then apply the construction mentioned to produce again an $L\mu$ formula $(\exists P.\tilde{\phi})^\vee$, which is true in any rooted structure whose ω-expansion satisfies $\exists P.\tilde{\phi}$; but if a structure satisfies ϕ, then its ω-expansion satisfies $\exists P.\tilde{\phi}$, since the original valuation of P provides a witness. With some more technical lemmas, it is shown that $(\exists P.\tilde{\phi})^\vee$ is indeed the uniform interpolant of ϕ with respect to the vocabulary omitting P, and this is the definition of $\widetilde{\exists} P.\phi$. A similar definition and construction also works for action labels: $\widetilde{\exists} a.\phi$ is the strongest implicate of ϕ in the language omitting the label a.

The reason for the notation $\widetilde{\exists} P.\phi$ is that from the construction, it can be seen that a rooted structure satisfies $\widetilde{\exists} P.\phi$ iff there is a bisimlation equivalent rooted structure in the vocabulary excluding P that satisfies ϕ.

In the above, $\widetilde{\exists} P.\phi$ was, by definition, an $L\mu$ formula. It is natural to ask whether bisimulation quantifiers can give the same expressive power as the fixpoint operators. It turns out to be not sufficient to add $\widetilde{\exists}$ to modal logic; but [16] does show that adding $\widetilde{\exists}$ to PDL gives $L\mu$.

The techniques used here also give further results. One of the most satisfying is a Lyndon theorem: if an $L\mu$ sentence is monotone in a proposition P, then it is equivalent to a sentence positive in P. The proof is intricate.

9 Generalized mu-calculi

We have seen that $L\mu$ has many nice properties. One interesting thread of research in recent years has been the investigation of why it enjoys these properties – is it because it is a *modal* fixpoint logic, because it is a *fixpoint* logic, or what else? In this section, we will briefly survey some of these investigations, and some of the more interesting generalizations of $L\mu$.

9.1 $L\mu$ with past

A simple extension of $L\mu$ is to include converse labels $\overline{a}$: $t \xrightarrow{\overline{a}} s$ iff $s \xrightarrow{a} t$. Modalities can now include converses. $L\mu$ with converse fails to have the finite model property: $\nu X.\langle a\rangle(X \wedge \mu Y.[\overline{a}]Y)$ is only satisfiable in an infinite state model. However, it retains both the tree model property and decidability of satisfiability (without an increase in complexity). The decidability proof uses two-way automata, alternating parity automata of section 5.3 whose modal language is extended with converse modalities [69].

9.2 Least fixpoint logic

Modal logic is a monadic fragment of first-order logic. $L\mu$ is such a fragment of least fixpoint logic, or LFP, obtained by adding fixpoint constructors to first order logic. It is primarily studied in the field of finite model theory; in the realm of infinite models, it is relatively little used, though occasionally used by set theorists as part of the theory of inductive definability. Finite model theorists use various notations, but usually do not use μ and ν, preferring to write LFP/GFP or **lfp**/**gfp**. We shall stick to a mu-calculus-like notation.

Assume the usual apparatus of first order logic over some structure S. *LFP* is obtained by adding *relation variables* $X, Y, \ldots$ of given arities, and a *least fixpoint operator* μ which forms *relation terms* $\mu X, \vec{x}.\phi$, where $\vec{x}$ is a tuple of arity(X) individual variables, and the relation variable X occurs only positively in ϕ. Assuming a valuation for the other free variables of ϕ, the semantics of $\mu X, \vec{x}.\phi$ is the least fixpoint of the map $S^n \to S^n$, where n is the arity of X, given by $T \mapsto \{\vec{x} : \phi[X := T]\}$.

LFP has the following properties (refer to a textbook such as [19] for proofs, and for details of results mentioned in this section without citations):

- On finite models with a built-in linear order, LFP captures polynomial time, which makes it useful for complexity theorists. (A logic L captures a complexity class C if every set in C can be defined by a formula of L, and conversely every L-definable set is in C.)
- On finite models, the fixpoint alternation hierarchy collapses, so that any LFP property can be expressed with a single fixpoint; provided that the arity of relation symbols is not bounded. If the arity is bounded, then the fixpoint hierarchy does not collapse.
- LFP does not have the finite model property.
- Satisfiability is undecidable.

LFP retains a fundamental semantic theorem which can be presented as a model-checking game as in section 4.4. The game is now played on an arena of formulae $\phi[s_1, \ldots, s_n]$ with elements s_i of the model for individual variables. The initial position is the starting closed formula ϕ_0 in positive normal form. $\forall$ is responsible for making a move from a position $(\phi \wedge \psi)[s_1, \ldots, s_n]$, the available choices are $\{\phi[s_1, \ldots, s_n], \psi[s_1, \ldots, s_n]\}$, and from a position $\forall x_{n+1}.\phi[s_1, \ldots, s_n]$ whose available choices are the set $\{\phi[s_1, \ldots, s_n, s] \mid s \in \mathfrak{S}\}$. $\exists$ is responsible for $\vee$ and existential quantification. Final positions are of the form $P[s_1, \ldots, s_n]$ and $\neg P[s_1, \ldots, s_n]$. $\exists$ wins such a position if it is true. Again, $\exists$ wins an infinite play if the outermost fixed point variable Y that occurs infinitely often in the play is a ν-variable. $\exists$ has a history-free winning strategy iff the initial formula is true of the structure.

9.3 Finite variable fixpoint logics

One of the topics studied in finite model theory is the *finite variable* fragments of FOL. These are the fragments FOL^k where the number of distinct variable names in a formula is restricted to a finite value k. Ordinary modal logic is obviously embeddable in FOL^2; there are several features of modal logic that are generalizable in some sense to FOL^2; and by adding certain operators to modal logic, one can regain FOL^2, albeit less succinctly [45]. Moreover, FOL^2 is reasonably tractable, and the decidability of modal logic follows from the decidability of FOL^2, which in turn follows from the fact that, like modal logic and $L\mu$, it enjoys the finite model property.

It is therefore natural to wonder if the good properties of modal mu-calculus might be explained by considering the finite variable fragments of LFP.

However, in a well-known paper ‘Why is modal logic so robustly decidable?’ [68], Vardi analysed the relationship between modal logic and FOL^2 more carefully, and argued that it does not adequately explain the good properties of modal logic. Furthermore, when one passes to the fixpoint version, it is even more inadequate: for example, although $L\mu$ is decidable, LFP^2 (and $L\mu^2$) is not decidable.

It appears, then, that finite variable fixpoint logics have little to say about $L\mu$. So what are the more useful related logics?

9.4 *Guarded fragments*

In [68], Vardi argued that the *tree model property* is responsible for the good behaviour of $L\mu$, and CTL. FOL^2 does not have this property. However, it turns out that there are fragments of FOL which do retain the tree model property or some suitable generalization of it. The discovery of these fragments needed a new perception of the characteristic features of modal logic seen as a fragment of FOL.

The fact that modal logic lies in FOL^2 is obvious. Somewhat less obvious is another property of the FO translations of modal logic formulae: *guardedness*. A FO quantification is *guarded* if it has the form $\forall\vec{y}.\alpha(\vec{x},\vec{y}) \Rightarrow \phi(\vec{x},\vec{y})$ or $\exists\vec{y}.\alpha(\vec{x},\vec{y}) \wedge \phi(\vec{x},\vec{y})$, where $\alpha(\ldots)$ is an atomic formula (i.e. α is a relation symbol or the equality symbol), and $\vec{x}$ includes all the free variables of ϕ. That is, when a quantified variable is introduced, its values must be connected by some relation to the values of the other variables mentioned in the formula. In the case of modal logic, the guards are the edge relations.

Guardedness was proposed by Andréka, van Benthem and Németi [2] as a better explanation of the robust decidability of modal logic. The guarded fragment GF of first-order logic has many of the nice properties of modal logic, for example

- GF is decidable.
- GF has the finite model property.
- GF has the appropriate generalization of the tree model property, namely that if a formula has a model, it has a model of 'bounded tree-width'. (Tree width is a graph-theoretic definition which measures how far a graph is from being a tree.)
- GF-equivalence can be characterized by a *guarded bisimulation*, as modal equivalence is characterized by bisimulation.

Grädel and Walukiewicz [29] studied the guarded fragment GFP of LFP. The syntactic formation rule for fixed points is: if $\phi(Y,\vec{x})$ is a guarded formula, Y occurs positively and not in the guards and all free variables of $\phi(Y,\vec{x})$ are contained in $\vec{x}$ then $\mu Y(\vec{x}).\phi(Y,\vec{x})$ is a formula of the guarded fragment of LFP. This fragment retains the tree model property but not the finite model property, making it a better meta-language for $L\mu$ than LFP^2. An interesting first result concerned the complexity: satisfiability for GFP is 2EXPTIME-complete. Grädel had earlier shown [27] that GF itself has 2EXPTIME satisfiability, so this is a situation where adding fixpoints does not increase complexity - a surprising result. However, it turns out that this depends on the unbounded *width* of formulae - the number of free variables in subformulae. If the width is bounded, then satisfiability drops to EXPTIME-complete, which agrees with that of $L\mu$. The decidability proof uses two-way alternating parity automata.

9.5 *Inflationary mu-calculus*

In finite model theory, as well as to some extent in classical definability theory, extensions of LFP have been studied which relax the requirement for the body of a fixpoint operator to be monotone. One such is *inflationary fixpoint logic* (IFP). In IFP, the semantics of the fixpoint operator (usually written **ifp** in the finite model theory literature, but here written μ_{I}) is modified. Rather than being defined as a fixpoint, it is defined in terms of approximants; and then at each approximant, the previous approximant is unioned in:

$$\|\mu_{\mathrm{I}} Z^{\alpha}.\phi\|^{\mathfrak{T}}_{\mathfrak{V}} = Z^{<\alpha} \cup \|\phi\|^{\mathfrak{T}}_{\mathfrak{V}[Z:=Z^{<\alpha}]}$$

On finite structures, IFP and LFP have long been known to be equi-expressive, and recently Kreutzer showed [39] that indeed they are equi-expressive on arbitrary structures. In [18] Dawar, Grädel and Kreutzer define inflationary modal mu-calculus, by using the above definition for fixpoints, and show that it is more powerful than $L\mu$, and complex in many ways. It does not have the finite model property, and it can express non-regular properties. Satisfiability is undecidable and even non-arithmetic, since it is possible to interpret arithmetic, by using the height of nodes in a well-founded tree as numbers. On the class of finite models, the increased power results in a model-checking complexity of PSPACE.

References

[1] H. R. Andersen, *Verification of Temporal Properties of Concurrent Systems*, DAIMI PB – 445, Computer Science Dept, Aarhus University, (1993).

[2] H. Andréka, J. van Benthem and I. Németi, Modal languages and bounded fragments of predicate logic. *J. Philos. Logic* **27**, 217–274 (1996).

[3] A. Arnold, The μ-calculus alternation hierarchy is strict on binary trees, *Theoretical Informatics and Applications* **33**(4/5) 329–340 (1999).

[4] A. Arnold and D. Niwinski, Fixed point characterization of Büchi automata on infinite trees, *J. Inf. Process. Cybern.* **EIK 26** 451–459 (1990).

[5] A. Arnold and D. Niwinski, *Rudiments of μ-Calculus*, Studies in Logic, Vol. **146**, North-Holland (2001).

[6] G. Bhat and R. Cleaveland, Efficient model checking via the equational mu-calculus, *Proc. LICS '96* 304–312 (1996).

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

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

[9] J. C. Bradfield, Simplifying the modal mu-calculus alternation hierarchy, *Proc. STACS 98* LNCS **1373** 39–49 (1998).

[10] J. C. Bradfield, Alternation on the binary tree, *Proc. 1st Int. Workshop on Fixpoints in Computer Science (FICS '98)* (ed. Z. Ésik), Masaryk Univ. Brno, 22–24 (1998).

[11] J. C. Bradfield, Fixpoints in arithmetic, transition systems and trees. *Theoretical Informatics and Applications*, **33**(4/5) 341–356 (1999).

[12] J. C. Bradfield and C. P. Stirling, Local model checking for infinite state spaces, *Theoret. Comput. Sci.* **96** 157–174, (1992).

[13] J. R. Büchi, On a decision method in restricted second order arithmetic. *Logic, Methodology and Philosophy of Science*, Proc. 1960 congress, Stanford Univ. Press, Stanford, CA, 1–11 (1962).

[14] E. M. Clarke and E. A. Emerson, Design and synthesis of synchronization skeletons using branching time temporal logic. LNCS **131** 52–71 (1981).

[15] E. M. Clarke, E. A. Emerson and A. P. Sistla, Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. on Programming Languages and Systems* **8** 244–263 (1986).

[16] G. d'Agostino and M. Hollenberg, Logical questions concerning the μ-calculus: interpolation, Lyndon and Łoś–Tarski. *J. Symbolic Logic* **65**(1) 310–332 (2000).

[17] M. Dam, CTL* and ECTL* as fragments of the modal mu-calculus. *Theoret. Comput. Sci.* **126** 77–96 (1994).

[18] A. Dawar, E. Grädel, and S. Kreutzer, Inflationary fixed points in modal logics, *ACM Trans. on Computational Logic* **5**(2) (2004).

[19] H.-D. Ebbinghaus and J. Flum, *Finite Model Theory*, 2nd edition, Springer, 1999.

[20] E. A. Emerson, Temporal and modal logic. *Handbook of Theoretical Computer Science*, Volume B., J. van Leeuwen Editor, Elsevier, 995–1072 (1990).

[21] E. A. Emerson and E. M. Clarke, Characterizing correctness properties of parallel programs using fixpoints. *Procs ICALP '80* LNCS **85** 169–181 (1980).

[22] E. A. Emerson and C. S. Jutla, The complexity of tree automata and logics of programs. In *Proc. 29th IEEE FOCS* 328–337 (1988).

[23] E. A. Emerson and C. S. Jutla, Tree automata, mu-calculus and determinacy. In *Proc. 32nd IEEE FOCS* 368–377 (1991).

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

[25] M. J. Fischer and R. E. Ladner, Propositional dynamic logic of regular programs. *J. Computer and System Science* **18** 194–211 (1979).

[26] R. Floyd, Assigning meanings to programs. In J. T. Schwartz, *Mathematical Aspects of Computer Science* 19–32. American Mathematical Society, Providence (1967).

[27] E. Grädel, On the restraining power of guards. *J. Symbol. Logic* **64**, 1719–1742 (1999).

[28] E. Grädel, W. Thomas and T. Wilke (Eds.), *Automata, Logics, and Infinite Games*, LNCS **2500** (2002).

[29] E. Grädel and I. Walukiewicz, Guarded fixed point logic, *Proc. 14th LICS* 45 (1999).

[30] Y. Gurevich and L. Harrington, Trees, automata and games. *Procs 14th ACM STOC* 60–65 (1982).

[31] M. Hennessy and R. Milner, On observing nondeterminism and concurrency, *Procs ICALP '80* LNCS **85** 295–309 (1980).

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

[33] D. Janin and I. Walukiewicz, On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. *Proc. CONCUR '96* LNCS **1119** 263–277 (1996).

[34] R. Kaivola, *Using Automata to Characterise Fixpoint Temporal Logics*. Ph.D. thesis, University of Edinburgh (1997).

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

[36] D. Kozen, A finite model theorem for the propositional μ-calculus. *Studia Logica* **47** 233–241 (1988).

[37] D. Kozen and R. Parikh, An elementary proof of the completeness of PDL. *Theoret. Comput. Sci.* **14** 113–118 (1981).

[38] D. Kozen and R. Parikh, A decision procedure for the propositional μ-calculus, LNCS **164** 313–325 (1984).

[39] S. Kreutzer, Expressive equivalence of least and inflationary fixed-point logic, *Annals of Pure and Applied Logic* **130**(1–3) 61–78 (2004).

[40] O. Kupferman, M. Vardi and P. Wolper, An automata-theoretic approach to branching-time model checking, *J. ACM* **42(2)** 312–360 (2000).

[41] R. Küsters and T. Wilke, Deciding the first level of the μ-calculus alternation hierarchy, *Proc. FSTTCS 2002*, LNCS **2556**, 241–252 (2002).

[42] L. Lamport, Proving the correctness of multiprocess programs. *IEEE Trans. Software Eng.* **2** 125-143 (1977).

[43] G. Lenzi, A hierarchy theorem for the mu-calculus. *Proc. ICALP '96* LNCS **1099** 87–109 (1996).

[44] X. Liu, C.R. Ramakrishnan, and S.A. Smolka, Fully local and efficient evaluation of alternating fixed points. *Proc. TACAS '98* LNCS **1384** 5–19 (1998).

[45] C. Lutz, U. Sattler and F. Wolter, Modal logic and the two-variable fragment, *Proc. CSL'01* (LNCS 2142), 247– (2001).

[46] A. Mader, *Verification of Modal Properties Using Boolean Equation Systems*. PhD Thesis, Technical University of Munich (1997).

[47] L. Maksimova, Absence of interpolation and of Beth's property in temporal logics with 'the next' operation. *Siberian Mathematical Journal* **32**(6) 109–113 (1991).

[48] Z. Manna and A. Pnueli, Formalization of properties of recursively defined functions. *Proc. ACM STOC* 201–210 (1969).

[49] Z. Manna and A. Pnueli, How to cook a temporal proof system for your pet language. *Proc. 10th ACM POPL* 141–154 (1983).

[50] R. Milner, *A Calculus of Communicating Systems*. LNCS **92** (1980).

[51] R. Milner, *Communication and Concurrency*. Prentice-Hall (1989).

[52] A. W. Mostowski, Regular expressions for infinite trees and a standard form of automata, *Computation Theory*, LNCS **208** 157–168 (1984).

[53] D. Niwiński, On fixed point clones, *Proc. ICALP '86* LNCS **226** 464–473 (1986).

[54] M. Otto, Bisimulation-invariant Ptime and higher-dimensional mu-calculus, *Theoret. Comput. Sci.* **224** 237–266 (1999).

[55] M. Otto, Eliminating recursion in the mu-calculus, *Proc. STACS '99*, LNCS **1563** 531–540 (1999).

[56] Park, D. Fixpoint induction and proofs of program properties. *Machine Intelligence*, **5**, 59-78, Edinburgh University Press (1969).

[57] V. Pratt, Semantical considerations of Floyd–Hoare logic. *Proc. 16th IEEE FOCS* 109–121 (1976).

[58] V. Pratt, A near-optimal method for reasoning about action. *Journal of Comput. and System Sciences* **20** 231-254 (1980).

[59] V. Pratt, A decidable mu-calculus, *Proc. 22nd IEEE FOCS*, 421-427 (1982).

[60] M. O. Rabin, Decidability of second-order theories and automata on infinite trees. *Transactions of American Mathematical Society*, **141**, 1-35 (1969).

[61] C. Stirling, Local model checking games. *Proc. Concur '95* LNCS **962** 1–11 (1995).

[62] C. Stirling and D. Walker, CCS, liveness, and local model checking in the linear time mu-calculus. *Proc. First International Workshop on Automatic Verification Methods for Finite State Systems*, LNCS **407** 166–178. (1990).

[63] C. Stirling and D. Walker, Local model checking in the modal mu-calculus. *Theor. Comput. Sci.* **89** 161–177. (1991).

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

[65] R. Streett, Propositional dynamic logic of looping and converse. *Proc. 13th ACM STOC* 375–383 (1981).

[66] W. Thomas, Languages, Automata, and Logic, in: *Handbook of Formal Language Theory* (G. Rozenberg, A. Salomaa, Eds.), Vol. III, Springer-Verlag , New York, pp. 389-455 (1998).

[67] M. Y. Vardi, A temporal fixpoint calculus. *Proc. 15th POPL*, 250–259 (1988).

[68] M. Y. Vardi, Why is modal logic so robustly decidable?, in *Descriptive Complexity and Finite Models*, N. Immerman and Ph. Kolaitis, eds., 1997, American Mathematical Society.

[69] M. Y. Vardi, Reasoning about the past with two-way automata, *Proc. ICALP '98* LNCS **1443** 628–641 (1998).

[70] M. Y. Vardi and P. Wolper, Yet another process logic. *Logics of Programs* LNCS **164** 501–512 (1983).

[71] I. Walukiewicz, Completeness of Kozen's axiomatisation of the propositional μ-calculus. *Information and Computation* **157**, 142–182 (2000).