

Recent Results on the Modal μ -Calculus: a Survey

GIACOMO LENZI

ABSTRACT. *Some theory of Modal μ -calculus is surveyed, with special emphasis on recent results.*

Keywords: Modal μ -Calculus, Kripke Models.
MS Classification 2010: 03B45, 03B44, 03B70

1. Introduction

In this survey we are interested in the Modal μ -Calculus. This formalism, introduced by Kozen in [42], is a powerful logic widely used in the area of specification and verification of computer systems, be they hardware or software. It lies at the basis of the powerful technique called model checking, see [22]. The importance of model checking in computer system verification is witnessed by the Turing Award given in 2007 to its main authors: E. Clarke, E. A. Emerson and J. Sifakis.

The Modal μ -calculus is obtained from modal logic by adding two operators μ and ν for the least and greatest fixpoints of monotone operators on sets.

We mention that a general theory of μ -calculi (not necessarily related to modal logic) is developed in [4].

Via Kripke semantics, the modal μ -calculus can be used to express properties of graphs. Intuitively, least fixpoints correspond to inductive definitions, and greatest fixpoints correspond to coinductive definitions. For instance, with least fixpoints one can express global liveness properties of a graph like “property P is true in some reachable point”, and with greatest fixpoints one expresses global safety properties of the kind “P is true in all reachable points”. These properties are not modally expressible (at least on arbitrary graphs) due to the local character of modal logic. Next, fixpoints can be nested, and by one nesting of least and greatest fixpoints we capture fairness properties like “P holds infinitely often”. Finally, with several nestings, one can express the existence of a winning strategy in a parity game.

Fixpoint alternation is also related to what is arguably the most important

open problem in the μ -calculus: the model checking problem. This problem has a solution in polynomial time for formulas with bounded fixpoint alternation depth. A general polynomial time algorithm is still out of reach. A substantial part of the ongoing research on the μ -calculus is related to this problem.

Another major theme is the expressiveness of μ -calculus and its fragments over various classes of graphs. It turns out that, on arbitrary graphs, the number of nestings between different fixpoints gives a strict infinite hierarchy. The situation may change if one considers special subclasses of graphs. A first, fundamental example is the class of transitive wellfounded graphs (aka the Gödel-Löb class or GL). From the celebrated de Jongh–Sambin Fixpoint Theorem, it follows that the μ -calculus in GL collapses to modal logic. That is, fixpoints in GL give no contribution to the expressiveness of μ -calculus.

The GL example shows that μ -calculus in subclasses of frames can be quite different from arbitrary frames. As we will see in this survey, many recent results on μ -calculus are focused on subclasses of frames.

Like all surveys, this one is not exhaustive. In particular, the choice of the material presented here depends on the personal taste of the author. Two predecessors of this survey are [19] and [20]. Here we concentrate (though not exclusively) on results of the last ten years approximately. In order to make the survey more accessible to non-experts, some more standard material is also added.

The author thanks the referees for their careful reading and their suggestions.

2. Preliminaries

2.1. Syntax

The formulas of Modal μ -Calculus are generated by the following grammar:

$$\phi ::= P \mid \neg P \mid X \mid \phi \wedge \phi' \mid \phi \vee \phi' \mid \Box \phi \mid \Diamond \phi \mid \mu X.\phi \mid \nu X.\phi,$$

where P ranges over an infinite set At of atoms and X ranges over an infinite set Var of variables. So, \neg is negation (applied only to atoms for convenience), \wedge and \vee are conjunction and disjunction, \Box and \Diamond are the universal and existential modal operators, μ is the least fixpoint operator, and ν is the greatest fixpoint operator.

Sometimes, especially in applications, it is useful to have several modal operators, corresponding to several relations in Kripke semantics. In this case, for each relation R there are a box operator $\Box_R \phi$ and a diamond operator $\Diamond_R \phi$.

Scopes of fixpoint variables, free and bound variables, etc. can be defined in the μ -calculus in analogy with variables of first order logic. Note that the formulas of the μ -calculus are not closed under negation. However, formulas

without free variables can be negated by swapping P and $\neg P$, \wedge and \vee , \square and \diamond , μ and ν .

Formulas of μ -calculus are closed under composition (also called replacement or substitution). If P is an atom and $\phi(P)$ is a formula containing P , we can replace P in ϕ with ψ as long as there is no capture, that is, no P occurs in the scope of a fixpoint operator μX or νX such that X is free in ψ . The result of the replacement is a formula of the μ -calculus denoted by $\phi(\psi)$.

2.2. Semantics

The semantics of Modal μ -Calculus is an extension of Kripke semantics for modal logic. We start with Kripke models $M = (V, R, \|P\|)$, where V is a set of vertices, R is a relation on V , and $\|P\| \subseteq V$ for every $P \in At \cup Var$. The structure $G = (V, R)$ is a graph, namely the underlying graph of the model M .

The semantics of a formula ϕ in a model M is a subset of V denoted by $\|\phi\|_M$, or $\|\phi\|$ if no confusion is possible. For the boolean operators we let $\|\neg P\| = V \setminus \|P\|$, $\|\phi \wedge \psi\| = \|\phi\| \cap \|\psi\|$, and $\|\phi \vee \psi\| = \|\phi\| \cup \|\psi\|$.

For the modal operators, $\|\diamond\phi\|$ is the set of all points which have some successor in $\|\phi\|$, and $\|\square\phi\|$ is the set of all points which have every successor in $\|\phi\|$.

Finally, the set $\|\mu X.\phi(X)\|$ is the least solution of the equation $E = \|\phi(E)\|$, where $\|\phi(E)\|$ is the semantics of ϕ in the model M modified so that $\|X\| = E$. The set $\|\nu X.\phi(X)\|$ is the greatest solution of the equation above.

If v is a vertex, we write also $M, v \models \phi$ if $v \in \|\phi\|_M$.

2.3. Examples

If P is an atom, the formula $\mu X.P \vee \diamond X$ expresses the fact that a point verifying property P is reachable. In fact, the set X of the points from where a P point is reachable verifies the equation $X = P \vee \diamond X$ and is the least solution of the equation. It may be convenient to call this formula *LIVENESS*(P).

Dually, the formula $\nu X.P \wedge \square X$ says that every reachable point verifies P . In fact, the set X of the points whose descendants satisfy P verifies the equation $X = P \wedge \square X$ and is the greatest solution. It may be convenient to call this formula *SAFETY*(P).

Also relevant for system verification is the formula $\mu X.\square X$, which says that the model has no infinite path. We call this formula *TERMINATION*.

To see nested fixpoints in action, consider the formula

$$\nu X.\mu Y.(P \wedge \diamond X) \vee \diamond Y.$$

Note that we can rewrite the formula as $\nu X.LIVENESS(P \wedge \diamond X)$.

This formula says that there is an infinite path where P occurs infinitely often. In fact, the set X of the points which see such a path verifies

$X = \text{LIVENESS}(P \wedge \diamond X)$ and is the greatest solution. We call the formula $\text{FAIRNESS}(P)$.

2.4. The syntactical fixpoint hierarchy

Fixpoint alternation depth is a measure of complexity of a formula. The levels of the alternation depth hierarchy Σ_n, Π_n are usually so defined to be closed under composition. Formally, one defines $\Sigma_0 = \Pi_0$ the set of all formulas without fixpoints. Then, the set Σ_{n+1} is the closure of $\Sigma_n \cup \Pi_n$ under composition and μ ; and dually, Π_{n+1} is the closure of $\Sigma_n \cup \Pi_n$ under composition and ν .

For instance, the formulas $\text{LIVENESS}(P)$ and TERMINATION are of level Σ_1 , the formula $\text{SAFETY}(P)$ is of level Π_1 , and the formula $\text{FAIRNESS}(P)$ is of level Π_2 .

Finally, Δ_n denotes the set $\Sigma_n \cap \Pi_n$. So, the fact that a property is Δ_n on a class of graphs means that it is both Σ_n and Π_n on that class.

The alternation depth of a formula ϕ is the smallest n such that $\phi \in \Delta_{n+1}$. This ensures that compositions of Σ_n and Π_n have alternation depth (at most) n .

2.5. Bisimulation

Recall that a bisimulation between two Kripke models M, N is a relation $B \subseteq V(M) \times V(N)$ such that, whenever vBw holds, we have:

- $v \models P$ if and only if $w \models P$ for every $P \in \text{At}$;
- if vRv' there is w' with wRw' and $v'Bw'$;
- symmetrically, if wRw' there is v' with vRv' and $v'Bw'$.

Two models with distinguished vertices (M, v) and (N, w) are called bisimilar if there is a bisimulation B between M and N such that vBw .

Bisimulation can be also restricted to a set S of atoms: bisimulation with respect to S is defined like bisimulation, but it is enough that $v \models P$ if and only if $w \models P$ happens for every $P \in S$.

2.6. Parity Games

Parity games are played on countable graphs, vertex-labeled with an alphabet of the form $\{E, O\} \times \{1, \dots, n\}$. The players are called Even and Odd. The game begins on an initial vertex v_0 . On E positions, Even moves, and on O positions, Odd moves. If either player cannot move, the other wins. In the case of an infinite play, Even wins if and only if the least index occurring infinitely often is even. Otherwise, Odd wins.

A strategy \mathcal{S} of a player Pl is a function from finite sequences of vertices v_0, v_1, \dots, v_k , where v_k is a Pl -vertex, to a successor of v_k . A strategy \mathcal{S} is winning if Pl wins all the plays that are induced by the strategy \mathcal{S} .

Since parity games are particular forms of Borel games in the sense of descriptive set theory, from Martin's Borel Determinacy Theorem [49] it follows that parity games are determined: there is always a player who has a winning strategy in the game. Actually we have the following strong form of determinacy (positional determinacy, see [27]): If a player Pl has a winning strategy in a parity game, then Pl has a positional winning strategy: that is, a winning strategy \mathcal{S} such that $\mathcal{S}(v_0, v_1, \dots, v_k)$ depends only on v_k . Note that in general, unlike parity games, Borel games are not positionally determined on all graphs.

Parity games are tightly related with the alternation depth hierarchy. If ϕ is a formula of level Σ_n , then the fact that $M, v \models \phi$ can be expressed by a parity game with n indices on a suitable graph. Conversely, the fact that Even has a winning strategy in a parity game with n indices can be expressed by a formula W_n , due to Walukiewicz, of level Σ_n . Namely:

$$W_n = \mu X_1. \nu X_2 \dots \theta X_n. \left(E \rightarrow \bigwedge_i (i \rightarrow X_i) \right) \wedge \left(O \rightarrow \bigwedge_i (i \rightarrow X_i) \right),$$

where $\theta = \mu$ if n is odd and $\theta = \nu$ if n is even, and where i ranges from 1 to n .

2.7. Parity automata

For surveys on automata see [67] and [51]. Tree automata were introduced in [59] as a “dynamic” counterpart of monadic second order logic over trees. Several equivalent variants of tree automata can be given, where the most important feature to choose is the acceptance condition. If we choose the parity acceptance condition, we obtain parity automata. We give a definition of them.

A parity automaton is a tuple $A = (Q, \Lambda, \delta, q_0, \Omega)$ where:

- Q is a finite set of states;
- Λ is a finite alphabet;
- $q_0 \in Q$ is the initial state;
- $\Omega : Q \rightarrow \omega$ is the priority function;
- $\delta : Q \times \Lambda \rightarrow Mod(Q)$, where $Mod(Q)$ is the set of all positive boolean combinations of $\square q$ and $\diamond q$, where q belongs to Q .

A semantic game (analogous to parity games) can be defined from an automaton A and a countable, pointed graph (G, v_0) equipped with a function $color : V \rightarrow \Lambda$.

The players are called Duplicator and Spoiler. Positions of the game are, alternately, elements of $Q \times V$ and subsets of $Q \times V$. Intuitively, Duplicator wants to build a sequence of states with infinitely many small even priorities, and Spoiler tries to build a sequence with infinitely many small odd priorities.

The initial position is (q_0, v_0) . On a position (q, v) , Duplicator chooses a “marking” relation $m \subseteq Q \times \text{succ}(v)$ with the following “correctness” property. Say that a successor w of v verifies a state q' in m if (w, q') belongs to m (and for completeness, assign no state to v unless v is a successor of itself). This makes the graph $\{v\} \cup \text{succ}(v)$ a Kripke structure K . We say that m is correct if this structure K verifies the modal formula $\delta(q, \text{color}(v))$.

Once Duplicator has chosen a correct marking m , Spoiler moves by choosing a pair $(q', v') \in m$, which becomes the current position, and so on.

If ever some player has no moves, the other wins. Otherwise, we have an infinite sequence of pairs (q, v) , and Duplicator wins if the least priority met infinitely often in the sequence is even; otherwise, Spoiler wins.

Since parity automata can be encoded as parity games, they enjoy positional determinacy. This is a good reason to choose parity automata rather than other, expressively equivalent kinds of automata.

μ -calculus and parity automata have the same expressive power. This allows one to reduce the satisfiability problem for the μ -calculus to the emptiness problem for tree automata, which is decidable, see e.g. [59]. This idea gives decision algorithms for the μ -calculus, among which an optimal one (running in exponential time) is in [27].

2.8. Particular Automata

We have seen that parity automata correspond to μ -calculus formulas. Some kinds of more restricted automata correspond to subclasses of μ -calculus formulas. Here are some examples.

A Büchi automaton is a parity automaton where $\Omega : Q \rightarrow \{0, 1\}$. In this case, Duplicator wins an infinite play if and only if the play meets zero states infinitely often. In the fixpoint hierarchy of the μ -calculus, Büchi automata correspond to the level Π_2 .

Dually, a coBüchi automaton is a parity automaton where $\Omega : Q \rightarrow \{1, 2\}$. This time, Duplicator wins an infinite play if and only if the play meets states of priority 2 always except for a finite number of times. In the fixpoint hierarchy of the μ -calculus, coBüchi automata correspond to the level Σ_2 .

A parity automaton is called weak if for every $(q, \lambda) \in Q \times \Lambda$ and every state q' occurring in $\delta(q, \lambda)$, we have $\Omega(q') \leq \Omega(q)$. So, along every transition, the priority does not increase. This implies that in every infinite play, the priority is eventually constant, and Duplicator wins if and only if this eventual priority is even. In the μ -calculus, weak automata correspond to compositions of formulas of level Σ_1 and Π_1 .

2.9. Tree Width

An important recent approach to computational complexity is fixed parameter complexity, which consists in studying classes of structures where a certain parameter is bounded, and trying to prove that intractable problems (e.g. NP-hard ones) become feasible when the parameter is bounded. Fixed parameter complexity works in the μ -calculus as well. A useful parameter for us (and for many other situations) is tree width of graphs, as we will see.

Intuitively, the tree width of a graph measures how far the graph is from being a tree. Intuitively, being close to a tree is a virtue, because many graph theoretic problems become much easier when restricted to trees. As a matter of fact, this analogy with trees does work, and many difficult graph problems become solvable in polynomial time on graphs with bounded tree width, see [12].

Moreover, tree width is useful in software verification because, as argued in [52], programs in many programming languages have control flow diagrams with low tree width (as long as no `goto` command or similar is used).

Formally, a tree decomposition of a finite graph G is a pair (\mathcal{X}, T) , where $\mathcal{X} = \{X_1, \dots, X_n\}$ is a family of subsets of V , and T is an undirected tree whose nodes are the subsets X_i , satisfying the following properties:

- The union of \mathcal{X} is V ;
- for every edge (v, w) in G , there is X_i containing both v and w ;
- if X_i and X_j both contain a vertex v , then all nodes X_z of the tree in the (unique) path between X_i and X_j contain v as well.

The width of a tree decomposition is the size of its largest node minus one. The tree width of a graph G is the minimum width among all possible tree decompositions of G .

For instance, trees have tree width 1, cycles of length 3 or more have tree width 2, and cliques with n vertices have tree width $n - 1$.

3. Complexity of μ -Calculus Formulas

3.1. The Semantical Fixpoint Hierarchy

It turns out that, on arbitrary graphs, the number of nestings between different fixpoints gives a strict infinite hierarchy, see [45], [14] and [3].

We note that the argument of [45] is purely syntactical and proves the infinity of the hierarchy only for the μ -calculus without negation. [14] builds upon a hierarchy theorem of [48] on fixpoint arithmetic. The proof has then been simplified in [15]. [16] and [17] expand on the relations between modal μ -calculus, fixpoint arithmetic and descriptive set theory.

The relation between μ -calculus and parity games is the key to the hierarchy result of [3]. In that paper, first it is shown that for every formula F of class Σ_n there is a contraction G_F on the complete metric space of all infinite binary trees, such that $T \models F$ if and only if $G_F(T) \models W_n$.

Now suppose that $\neg W_n$ is equivalent to a formula F of class Σ_n , and consider the fixpoint T_F of the contraction G_F . Then, $T_F \models F$ if and only if $G_F(T_F) \models W_n$, hence $T_F \models \neg W_n$ if and only if $T_F \models W_n$, a contradiction. Therefore, W_n is not Π_n for every n , and the hierarchy is infinite.

The paper [3] settles the infinity of the hierarchy on arbitrary graphs. From this one derives also the infinity of the hierarchy on finite graphs, because the μ -Calculus has the finite model property: every formula which is true in some graph is true in some finite graph, see [65].

Another point concerning the fixpoint hierarchy is the status of the intermediate classes Δ_n , also called ambiguous classes. In particular the question is whether Δ_n coincides with compositions of formulas of class Σ_{n-1} and Π_{n-1} . Clearly every such composition belongs to Δ_n (in any class of graphs). For the converse, [43] shows, with a topological argument, that every Δ_1 property is expressible in modal logic (i.e. in the μ -calculus without fixpoints), and [44] shows that Δ_2 coincides with compositions of Π_1 and Σ_1 formulas. This fragment of the μ -calculus is sometimes called the alternation-free fragment. Then, somewhat unexpectedly, [5] shows that for $n > 2$, Δ_n and compositions of Σ_{n-1} and Π_{n-1} differ. The argument is similar to the contraction argument of [3].

3.2. The Hierarchy in some Subclasses

For proper subclasses of graphs, in general, the contraction argument of [3] simply does not go through, and the semantical fixpoint hierarchy may collapse or not. Here are some examples.

In [2], the class GL (see the introduction) is considered. Probably this is one of the most studied subclasses of graphs, from the Seventies on, in view of its relation with Gödel theorems and the logic of provability in Peano Arithmetic. The authors of [2] reprove the classical De Jongh-Sambin Theorem, which states that every guarded formula has a fixpoint in GL. Uniqueness of the fixpoint was proved later by Bernardi, de Jongh and Sambin independently (see [63]). The authors extend the syntax of the μ -calculus to a μ^\sim -calculus where fixpoint variables can appear everywhere in the formula, also under negations, as long as they are under modalities. The semantics of the μ^\sim -calculus in GL is given via an evaluation game. By means of this game, the authors show that the μ^\sim -calculus collapses to the modal fragment and provide an explicit translation. This allows them to compute the fixpoints involved in the de Jongh-Sambin Theorem. The collapse of μ -calculus in GL was already proved, using the de Jongh-Sambin Theorem, by [69] and [70].

Another example is given by transitive graphs, or in modal terms, the class $K4$. This is clearly an interesting class: for instance, the relation “event A is posterior to event B” is transitive. There are several papers on the status of the hierarchy on transitive graphs. Probably the first is [46], whose proof contains an error corrected in [24]. The upper bound of [46] is that in $K4$, the μ -calculus is included in Büchi automata, that is, the level Π_2 of the alternation hierarchy. It turns out that one can do slightly better: in $K4$, the μ -calculus is included in compositions of Σ_1 and Π_1 . This is proven independently in [24], [25] and [1]. In particular, [1] perform an analysis of evaluation games in transitive graphs. Their main technical result is that if a variable x of a formula $\phi(x)$ occurs only once and is in the scope of a \square , then in $K4$ we have $\nu x.\phi(x) = \phi(\phi(true))$.

[1] considers also transitive symmetric graphs, where the collapse property is even stronger: the μ -calculus coincides with modal logic. Here the equality $\nu x.\phi(x) = \phi(\phi(true))$ holds identically. As a matter of fact, this proves that in $S5$ (the class of all equivalence graphs), the μ -calculus collapses to modal logic.

A subclass where the collapse does not happen is T , the class of all reflexive frames. This is again proven in [1] via game theoretic methods.

A quite trivial class, from the viewpoint of fixpoint alternation, is given by finite trees. It is not difficult to see that on this class, the μ -calculus collapses to Δ_1 (in fact, $\mu X.\phi(X)$ always coincides with $\nu X.\phi(X)$) but not to modal logic. However, finite trees pose interesting problems in many other respects, also in μ -calculus theory: for instance, [66] shows the completeness of a natural proof system for the μ -calculus over finite trees.

In the same vein, a possible theme for further research is the hierarchy on graphs of bounded tree width. It would be interesting to see if one can come up with an upper bound on the alternation depth of the μ -calculus on graphs of bounded tree width.

3.3. The Variable Hierarchy

Since model checking is polynomial time computable when the fixpoint alternation is bounded, we can probably say that fixpoint alternation is an important measure of complexity of μ -calculus formulas. However it is not the only one we can imagine. In descriptive complexity, a well known measure of complexity, for instance in first order logic, is the number of variables of a formula. This number, in descriptive complexity, is related to space complexity, see [35]. The equivalent measure in μ -calculus is the number of fixpoint variables. It is intended that each variable can be declared several times in a formula, so for instance the formula

$$\mu X.\diamond X \vee \mu X.\square X \vee \nu X.\diamond X \vee \nu Y.\square\diamond Y$$

has two variables X and Y .

The results of [9], [11] and [10] show that the variable hierarchy of the μ -calculus is infinite: for every n , there is a formula which is not expressible with less than n variables.

The examples of formulas which require arbitrarily many variables are concerned with description of a given finite Kripke structure. Specifically, we identify a parameter of directed graphs, called entanglement, which measures how many variables are sufficient to describe, up to bisimulation, any Kripke structure over that graph. We prove that every directed graph of entanglement k can be turned into a Kripke structure that cannot be described with fewer than k variables.

The proof of the hierarchy theorem in [10] consists of two main parts. First, as in [9], the strictness of the hierarchy is established for the case of existential formulas, i.e., formulas built without using universal modalities. It is shown that no existential formula with fewer than k variables can define the simulation type of a Kripke structure of entanglement k , under a particular labelling (recall that simulation is a kind of “one-sided” bisimulation). Entanglement of a graph can be defined in combinatorial terms, but it can be also nicely characterized with the following cop-robber game.

There are a robber and k cops. At the beginning, the robber is at the given initial position u of G and the cops are outside the graph. In any round, the cops may either stay where they are or place one of themselves on the current position v of the robber. The latter, in turn, has to move to a successor w of v that is not occupied by any cop. If no such position exists, the robber is caught and the cops have won. Note that the robber sees the move of the cops before he decides on his own move, and he is forced to leave his current position, regardless of whether the cops move or not. Now the minimal number k such that k cops have a strategy to catch the robber on a graph G starting with a node u is called the entanglement of (G, u) .

In the second part of [10], we have a preservation theorem stating that every formula defining the simulation type of a strongly connected structure can be transformed into an existential formula without increasing the number of variables. Thus the strictness of the variable hierarchy for the full μ -calculus follows from its strictness in the existential case.

Entanglement has been subsequently studied, for instance in [61], and successfully applied to variable hierarchy results in other μ -calculi (in the sense of [4]).

4. Fragments and Sublogics

4.1. Continuous μ -Calculus

A natural fragment of μ -calculus is the continuous one. A formula $\phi(P)$ is continuous in an atom P if it is monotonic in atom P , and $\phi(P)$ implies that

there is $Q \subseteq P$ such that Q is finite and $\phi(Q)$ holds. This fragment is studied in [30]. First, it is shown that continuity corresponds to the formulas built using the operators \wedge, \vee, \diamond and μ . Second, it is shown to be decidable whether a formula is continuous in P .

A related notion is constructivity. The constructive formulas are the formulas whose fixpoint is reached in at most ω steps. Formally, a sentence $\phi(p)$ is constructive in p if the least fixpoint of the semantic map Sem_ϕ sending the set E to $\|\phi(E)\|$ is equal to the union of all finite iterations $Sem_\phi^i(\emptyset)$.

It is folklore that if a formula is continuous, then it is constructive. The other implication does not hold in general. A counterexample is $\phi(p) = \nu X.p \wedge \diamond X$, a formula saying that there is an infinite path where a property p holds always. The formula is constructive because $\phi(\emptyset) = \emptyset$; on the other hand, it is not continuous because it is true in an infinite linear model where p holds always, but it is false in any infinite linear model where p holds finitely often.

However, interesting questions concerning the link between constructivity and continuity remain. One due to Venema is the following: given a constructive formula ϕ , does there exist a continuous formula ψ such that $\mu X.\phi = \mu X.\psi$?

Other interesting links exist between continuity and the program logic PDL (Propositional Dynamic Logic), which can be considered as a sublogic of the μ -calculus, see below.

4.2. Temporal Sublogics

It is notoriously hard to understand the meanings of μ -calculus formulas, especially when fixpoints are nested. However, there are several sublogics of the μ -calculus which are more transparent and still quite expressive; so, if we need to verify a property of a system, we can first express it in a sublogic and then translate it in the μ -calculus.

Sublogics of the μ -calculus include temporal logics, program logics and game logics. In this subsection we briefly recall the temporal logics CTL and CTL^* , and in the next we recall the program logic PDL , and Parikh's game logic PGL (we use the name PGL rather than GL to avoid confusion with Gödel-Löb Logic).

First, the logics CTL and CTL^* are intended to reason about paths in a tree model, which represent computations of a system. Every formula can hold or not along a path; some formulas make sense also on nodes of the tree, and are called node formulas.

The temporal operators available in these logics are the next operator X and the until operator U . A path π satisfies $X\phi$ if the subpath starting with the second point of π verifies ϕ ; and π satisfies $\phi U \psi$ if some subpath satisfies ψ , and all subpaths inbetween verify ϕ .

Useful abbreviations are $F\phi = trueU\phi$ and $G\phi = \neg F\neg\phi$. So, a path satisfies $F\phi$ if ϕ is true in some subpath, and satisfies $G\phi$ if ϕ is true in every subpath.

Moreover, we have the existential and universal path quantifiers E and A . If ϕ is a formula, then $E\phi$ is a node formula which holds on a node v if some path starting with v satisfies ϕ , and similarly for $A\phi$.

In CTL^* , all the temporal operators and quantifiers can be used freely, whereas in CTL , the combinations allowed are only $EX\phi$, $AX\phi$, $E(\phi U\psi)$ and $A(\phi U\psi)$.

Note that CTL is quite simpler than CTL^* : for the μ -calculus translation, the former can be translated in compositions of Σ_1 and Π_1 , whereas for the latter we need compositions of Σ_2 and Π_2 .

For more on CTL and CTL^* see [26].

4.3. Program and Game Sublogics

The propositional dynamic logic PDL is intended to reason about programs, so it has two sorts of syntactic objects, programs and formulas, defined by simultaneous induction. Starting with atomic programs, one can perform union and concatenation of two programs, and finite iteration (Kleene star) of a program; moreover, if ϕ is a formula, the program denoted by $\phi?$ performs a test on whether the current state satisfies ϕ . Formulas are given by multimodal logic over the programs (so, there is a modal operator \diamond_π for every program π). For more on PDL see [58] and [29].

Parikh's game logic PGL , see [57], is a logic for two player games, seen as generalizations of programs. The syntax of PGL is obtained from PDL by adding a dualizing operator d on games, whose meaning is to invert the role of the two players.

For the convenience of the reader, we sketch formal syntax and semantics of PGL . Formulas ϕ and games γ are defined by the syntax:

$$\begin{aligned}\phi &::= p \mid \neg\phi \mid \phi \vee \phi' \mid \diamond_\gamma\phi \\ \gamma &::= g \mid \phi? \mid \gamma; \gamma' \mid \gamma \cup \gamma' \mid \gamma^* \mid \gamma^d,\end{aligned}$$

where p ranges over a set P_0 of atomic propositions, and g ranges over a set Γ_0 of atomic games.

The semantics of PGL can be given in terms of Angel-Demon games, to be played on tuples $M = (S, V_p, E_g)$, where S is a set, $V_p \subseteq S$ for every $p \in P_0$, and $E_g : S \rightarrow P(P(S))$ for every $g \in \Gamma_0$. So we have Kripke structures together with a map saying what sets can be reached from a given atomic game g starting with a vertex $s \in S$.

Let us write $sE_\gamma X$ for $X \in E_\gamma(s)$. Intuitively, this means that Angel can reach the set X starting with s and playing the game γ . Let us also write $E_\gamma(Y) = \{s \in S \mid sE_\gamma Y\}$.

The semantics of a formula in a model, denoted $\|\phi\|$, and the effectivity map of a game in the model, denoted by E_γ , are defined inductively as follows. The boolean clauses are as usual; the diamond clause is that $s \in \|\diamond_\gamma \phi\|$ if and only if $sE_\gamma\|\phi\|$; for concatenation we have $E_{\alpha;\beta}(Y) = E_\alpha(E_\beta(Y))$; for union we have $E_{\alpha\cup\beta}(Y) = E_\alpha(Y) \cup E_\beta(Y)$; for test we have $E_{\phi?}(Y) = \|\phi\| \cap Y$; for Kleene star we have $E_{\alpha^*}(Y) = \mu X.Y \cup E_\alpha(X)$; and finally, for duality we have $E_{\alpha^d}(Y) = \overline{E_\alpha(\overline{Y})}$, where the horizontal bar denotes complement.

It can be shown that *PGL* subsumes *CTL** and the Walukiewicz game formulas W_n , and that *PGL* can be translated into the μ -calculus with two variables, see [6]. But the variable hierarchy of the μ -calculus is infinite, so *PGL* is a proper fragment of the μ -calculus, and this result solves an open question of [56].

5. Modal Logics, μ -Calculus and Bisimulation

5.1. Bisimulation Invariance

Like modal logic, the μ -calculus is invariant under bisimulation. In fact, μ -calculus itself can be viewed as an infinitary modal logic, provided one defines the semantics inductively via ordinal approximants.

Since every model M is bisimilar to its unfolding, which is a tree whose vertices are the finite paths in M with a natural structure of a model, it follows that every formula which has a model has a tree model as well.

Another observation is that the μ -calculus can be translated into monadic second order logic. In fact, a vertex v verifies $\mu X.F(X)$ if and only if v belongs to every set E such that $E = F(E)$; and v verifies $\nu X.F(X)$ if and only if v belongs to some set E such that $E = F(E)$.

An interesting converse direction, with respect to the previous two observations, has been proven in [38]: every formula of monadic second order logic which is invariant under bisimulation is equivalent to a formula of the μ -calculus. The key point of the proof is that on trees, monadic second order logic can be translated into tree automata.

Note that a similar result was achieved for first order logic by [68]: modal logic is exactly the fragment of first order logic which is invariant under bisimulation.

The [68] result was specialized to finite graphs by [60]. The parallel result for the μ -calculus was settled in [25] together with quite a lot of results for subclasses of frames. In particular, an unexpected outcome of [25] is concerned with finite transitive graphs: in these graphs the bisimulation invariant fragments of first order logic and monadic second order logic coincide, and they are larger than modal logic.

5.2. Deciding Modality

In studying expressiveness of logics or their fragments, an interesting question is the following: given a larger logic and a smaller logic, decide whether a formula in the larger logic is equivalent (with respect to a given class of structures) to one in the smaller. This question may be difficult because it requires insight in the structure of the sublogic. For instance in the μ -calculus, a natural candidate sublogic is modal logic. An algorithm to decide modality of a μ -calculus formula over arbitrary graphs was given in [55]. The idea was that a modal formula looks only at the first few levels of a tree.

The same question was answered for transitive graphs in [13]. Equivalently, the problem is to characterize regular tree languages definable in modal logic, where modalities refer to the transitive closure of the tree (usually these modalities are denoted by \Box^+ and \Diamond^+).

In this breakthrough paper, notions of forest algebra (with the operations of disjoint union and grafting) and regular forest language are introduced, so that μ -definable tree languages are particular cases. Then, among regular forest languages, modally definable forest languages are characterized by bisimulation invariance plus a suitable equation. Finally, both bisimulation invariance and the equation (actually any possible equation in the algebra) are then shown to be decidable.

5.3. Uniform Interpolation

A logic satisfies interpolation if whenever ϕ and ψ are formulas with $\phi \models \psi$ (that is, ψ is a consequence of ϕ), there is an “interpolant” formula χ in the common language of ϕ and ψ such that $\phi \models \chi$ and $\chi \models \psi$.

Since Craig’s proof of the interpolation for first order logic, it became natural, once a logic is introduced, to ask whether it satisfies the same property. In [23] a strong form of interpolation is proved for μ -calculus: uniform interpolation. That is, the μ -calculus satisfies the following: if ϕ is a formula and L is a language (i.e. a set of atoms), there is a “uniform interpolant” formula $I(\phi, L)$ in L such that:

- $\phi \models I(\phi, L)$;
- for every formula ψ such that $L(\psi) \cap L(\phi) \subseteq L$, if $\phi \models \psi$ then $I(\phi, L) \models \psi$.

In this sense the μ -calculus is better behaved than first order logic (which has ordinary interpolation but not the uniform one) and monadic second order logic (which does not even have the ordinary version).

The key tool in [23] are bisimulation quantifiers. Recall that the existential bisimulation quantifier, denoted $\exists P.\phi$, has the following meaning: a model M satisfies $\exists P.\phi$ if there is a model N , bisimilar to M on all atoms except for

the atom P , which verifies ϕ . [23] shows, with automata theoretic methods, that the μ -calculus is closed under bisimulation quantifiers. Then $I(\phi, L)$ is obtained from ϕ by quantifying existentially over the atoms which are not in L .

6. Axiomatization

6.1. Axioms for the Full μ -Calculus

In the paper where the modal μ -calculus is presented, [42], we have a very natural Hilbert style axiom system for the logic. The system is given by the system K for modal logic plus:

- the axiom schema $\alpha(\mu X.\alpha(X)) \rightarrow \mu X.\alpha(X)$;
- the rule: from $\alpha(\phi) \rightarrow \phi$ infer $\mu X.\alpha(X) \rightarrow \phi$.

In [42], completeness of the system is proved only for aconjunctive formulas, a class of formulas where conjunction is restricted. Full completeness is proved in [72], where the key point is to reduce to a further subclass of aconjunctive formulas, called the disjunctive formulas. Recall that disjunctive formulas, introduced in [37], are generated by the following rules:

- P , $\neg P$ and X are disjunctive;
- if α, β are disjunctive then $\alpha \vee \beta$ is disjunctive;
- if α is a finite conjunction of atoms and negated atoms, and β_1, \dots, β_n are disjunctive, then $\alpha \wedge \square(\beta_1 \vee \dots \vee \beta_n) \wedge \diamond\beta_1 \wedge \dots \wedge \diamond\beta_n$ is disjunctive;
- if ϕ is disjunctive then also $\mu X.\phi$ and $\nu X.\phi$ are disjunctive.

It is still an open problem to find good cut-free proof systems for the modal μ -calculus (this is not a completely precise question, since “cut free” here means only that no modus ponens or similar rule is necessary for the system to work). One difficulty is that intuitively, whereas a proof of $\mu X.\phi$ should be something “wellfounded”, a proof of $\nu X.\phi$ should be allowed to be “circular”. However, good proof systems do exist for related formalisms (μ -calculi in the sense of [4]), see [61].

6.2. Common Knowledge

Common knowledge is an epistemic concept useful in many applications. Like epistemic logic can be viewed as a multimodal logic (where a modality $K_i\phi$ means that agent i knows the fact ϕ), common knowledge can be viewed as a property expressible in a multimodal μ -calculus: if we have n agents, then

the fact that ϕ is common knowledge among the agents can be expressed by a greatest fixpoint

$$C\phi = \nu X. \phi \wedge K_1 X \wedge \dots \wedge K_n X.$$

Gerhard Jaeger and his school have been studying proof systems for common knowledge, and in particular, cut-free sequent systems. In [36] they find such a system. They start with a natural infinitary system including an ω -rule, and then they manage to replace the ω -rule with an infinite (but simple) set of finitary rules.

7. Model Checking

The model checking problem for the μ -calculus is the following: given a finite Kripke model M , a point v of M and a formula ϕ , decide if $M, v \models \phi$. It can be seen that this problem is polynomial time equivalent to the problem of solving a parity game.

It is known that the two problems are in the complexity class UP (standing for Unique P), that is, the problems solvable in polynomial time by a nondeterministic Turing machine having at most one accepting computation on each input, see [39]. Note that UP is a class somewhere between P and NP , and a $co-UP$ bound follows by complementation.

Several algorithms have been proposed, starting with the first model checking algorithm of [28]; the working time of this algorithm is $O(m \cdot n^{d+1})$, where m is the size of ϕ , n is the size of M and d is the alternation depth of ϕ .

Subsequently, [47] improved the complexity of the Emerson-Lei algorithm to $O(m \cdot n^{\lfloor d/2 \rfloor + 1})$; and as one referee points out, [62] further improved the bound to $O(m \cdot n^{d/3})$ by reducing to parity games and combining ideas from [40] and [41]. In turn, the algorithm of [40] has approximately the same time complexity of [28], but features linear space complexity.

There also exists a polynomial time model checking algorithm on graphs of bounded tree width, see [52]. Recall that Courcelle's theory of monadic second order logic implies that on graphs of bounded tree width k , the model checking problem can be solved in linear time in the size of the graph, that is, the time complexity is $O(n)$. However, the constant hidden in the O (depending on the formula and on the tree width) is large according to Courcelle's bound. [52] manages to reduce the complexity to $O(n \cdot (km)^2 \cdot d^{2((k+1)m)^2})$, so a little more than exponential in dkm .

Intuitively, the idea of [52] for solving parity games is first to solve locally the game in every node of the tree decomposition, and then to combine the local solutions in a bottom up manner. This use of the tree decomposition works for many other problems, see [12].

Like for tree width, feasibility results for parity games have been obtained for other analogous measures, for instance, bounded entanglement [8],

bounded DAG-width, see [7] and [53], bounded Kelly-width [34], or bounded clique width [54].

For the general case, the best we have so far is a subexponential algorithm ($n^{O(\sqrt{n})}$), see [41], building on algorithms of McNaughton [50] and Zielonka, see [73] and [33]. A general polynomial algorithm is actively searched.

Despite the unsatisfactory asymptotic bounds, however, model checking tools exist, see [32] for a recent one, and also (as one referee suggests) practical model checking algorithms such as [71]. As a matter of fact, it has been a hard task, done in [31], to show that the algorithm of [71] is *not* efficient, and Oliver Friedmann received a Kleene award for this.

We mention that the model checking problem for particular classes of finitely presented, infinite structures is treated, for instance, in [18] and [64]. In this case one uses a proof system rather than an algorithm, and a property holds of a system if and only if there is a proof that this is the case. In particular, [64] considers a mutual exclusion algorithm modeled as a CCS process, and [18] treats a Petri net example.

One referee adds the following remarks on infinite structures. Since the μ -calculus is a fragment of Monadic Second Order Logic, μ -calculus model checking is decidable whenever MSO model checking is so, see e.g. the Caucal hierarchy, a class of infinite graphs obtained from finite graphs via unfoldings and inverse rational mappings. Interestingly, there are structures where μ -calculus model checking is decidable, but MSO model checking is not, see e.g. [21]; and there are structures (e.g. grids) with decidable μ -calculus model checking, whose version with back edges has an undecidable μ -calculus model checking (in the case of grids, this is obtained by reduction from the termination problem of two counter machines).

REFERENCES

- [1] L. ALBERUCCI AND A. FACCHINI, *The modal μ -calculus hierarchy over restricted classes of transition systems*, J. Symb. Logic **74** (2009), 1367–1400.
- [2] L. ALBERUCCI AND A. FACCHINI, *On modal μ -calculus and Gödel-Löb logic*, Studia Logica **91** (2009), 145–169.
- [3] A. ARNOLD, *The μ -calculus alternation-depth hierarchy is strict on binary trees*, Inf. Théor. Appl. **33** (1999), 329–340.
- [4] A. ARNOLD AND D. NIWINSKI, *Rudiments of μ -calculus*, Studies in Logic and the Foundations of Mathematics volume 146, North-Holland, Amsterdam (2001).
- [5] A. ARNOLD AND L. SANTOCANALE, *Ambiguous classes in μ -calculi hierarchies*, Theoret. Comput. Sci. **333** (2005), 265–296.
- [6] D. BERWANGER, *Game logic is strong enough for parity games*, Studia Logica **75** (2003), 205–219.
- [7] D. BERWANGER, A. DAWAR, P. HUNTER AND S. KREUTZER, *Dag-width and parity games*, LNCS volume 3848, Springer, Berlin (2006), 524–536.

- [8] D. BERWANGER AND E. GRÄDEL, *Entanglement — A measure for the complexity of directed graphs with applications to logic and games*, LNCS volume 3452, Springer, Berlin (2005), 209–233.
- [9] D. BERWANGER, E. GRÄDEL AND G. LENZI, *On the variable hierarchy of the modal μ -calculus*, LNCS volume 2471, Springer, Berlin (2002), 352–366.
- [10] D. BERWANGER, E. GRÄDEL AND G. LENZI, *The variable hierarchy of the μ -calculus is strict*, Theor Comput. Syst. **40** (2007), 437–466.
- [11] D. BERWANGER AND G. LENZI, *The variable hierarchy of the μ -calculus is strict*, LNCS volume 3404, Springer, Berlin (2005), 97–109.
- [12] H. BODLAENDER, *Treewidth: Algorithmic techniques and results*, LNCS volume 1295, Springer, Berlin (1997), 19–36.
- [13] M. BOJANCZYK AND T. IDZIASZEK, *Algebra for infinite forests with an application to the temporal logic ef* , LNCS volume 5710, Springer, Berlin (2009), 131–145.
- [14] J. BRADFIELD, *The modal μ -calculus alternation hierarchy is strict*, Theoret. Comput. Sci. **195** (1998), 133–153.
- [15] J. BRADFIELD, *Simplifying the modal μ -calculus alternation hierarchy*, LNCS volume 1373, Springer, Berlin (1998), 39–49.
- [16] J. BRADFIELD, *Fixpoints, games and the difference hierarchy*, Theor. Inform. Appl. **37** (2003), 1–15.
- [17] J. BRADFIELD, J. DUPARC AND S. QUICKERT, *Transfinite extension of the μ -calculus*, LNCS volume 3634, Springer, Berlin (2005), 384–396.
- [18] J. BRADFIELD AND C. STIRLING, *Local model checking for infinite state spaces*, Theoret. Comput. Sci. **96** (1992), 157–174.
- [19] J. BRADFIELD AND C. STIRLING, *Modal logics and μ -calculi: an introduction*, in J. BERGSTRA, A. PONSE AND S. SMOLKA, *Handbook of process algebra*, Elsevier, U.S.A. (2001), 293–330.
- [20] J. BRADFIELD AND C. STIRLING, *Modal μ -calculi*, in P. BLACKBURN, J. VAN BENTHEM AND F. WOLTER, *The handbook of modal logic*, Elsevier, U.S.A. (2006), 721–756.
- [21] C. BROADBENT AND L. ONG, *On global model checking trees generated by higher-order recursion schemes*, LNCS volume 5504, Springer, Berlin (2009), 107–121.
- [22] E. CLARKE, O. GRUMBERG AND D. PELED, *Model checking*, MIT Press, U.S.A. (2000).
- [23] G. D’AGOSTINO AND M. HOLLENBERG, *Logical questions concerning the μ -calculus: interpolation, Lyndon and Los-Tarski*, J. Symb. Logic **65** (2000), 310–332.
- [24] G. D’AGOSTINO AND G. LENZI, *On the μ -calculus over transitive and finite transitive frames*, accepted on Theoret. Comput. Sci.
- [25] A. DAWAR AND M. OTTO, *Modal characterisation theorems over special classes of frames*, Ann. Pure Appl. Logic **161** (2009), 1–42.
- [26] E.A. EMERSON, *Temporal and modal logic*, in J. VAN LEEUWEN, *Handbook of theoretical computer science. Volume B: Formal models and semantics*, North-Holland Pub. Co./MIT Press, (1990), 995–1072.
- [27] E.A. EMERSON AND C.S. JUTLA, *Tree automata, μ -calculus and determinacy*, in *IEEE Proceedings on Foundations of Computer Science (FOCS 1991)*, IEEE

- Computer Society Press, U.S.A. (1991), 368–377.
- [28] E.A. EMERSON AND C.L. LEI, *Efficient model checking in fragments of the propositional μ -calculus*, in *IEEE Symposium on Logic in Computer Science (LICS 1986)*, IEEE Computer Society Press, U.S.A. (1986), 267–278.
 - [29] M.J. FISCHER AND R.E. LADNER, *Propositional dynamic logic of regular programs*, *J. Comput. Syst. Sci.* **18** (1979), 194–211.
 - [30] G. FONTAINE, *Continuous fragment of the μ -calculus*, LNCS volume 5213, Springer, Berlin (2008), 139–153.
 - [31] O. FRIEDMANN, *An exponential lower bound for the parity game strategy improvement algorithm as we know it*, in *IEEE Symposium on Logic In Computer Science (LICS 2009)*, IEEE Computer Society Press, U.S.A. (2009), 145–156.
 - [32] O. FRIEDMANN AND M. LANGE, *A solver for modal fixpoint logics*, *Electr. Notes Theor. Comput. Sci.* **262** (2010), 99–111.
 - [33] E. GRÄDEL, W. THOMAS AND T. WILKE, *Automata, logics, and infinite games. a guide to current research*, LNCS volume 2500, Springer, Berlin (2002).
 - [34] P. HUNTER, *Complexity and infinite games on finite graphs*, Ph.D. thesis, University of Cambridge, Cambridge (2007).
 - [35] N. IMMERMANN, J.F. BUSS AND D.A. MIX BARRINGTON, *Number of variables is equivalent to space*, *J. Symb. Logic* **66** (2001), 1217–1230.
 - [36] G. JÄGER, M. KRETZ, AND T. STUDER, *Cut-free common knowledge*, *J. Appl. Logic* **5** (2007), 681–689.
 - [37] D. JANIN AND I. WALUKIEWICZ, *Automata for the modal μ -calculus and related results*, LNCS volume 969, Springer, Berlin (1995), 552–562.
 - [38] D. JANIN AND I. WALUKIEWICZ, *On the expressive completeness of the propositional μ -calculus w.r.t. monadic second-order logic*, LNCS volume 1119, Springer, Berlin (1995), 263–277.
 - [39] M. JURDZINSKI, *Deciding the winner in parity games is in $up \cap co-up$* , *Inform. Process. Lett.* **68** (1998), 119–124.
 - [40] M. JURDZINSKI, *Small progress measures for solving parity games*, LNCS volume 1770, Springer, Berlin (2000), 290–301.
 - [41] M. JURDZINSKI, M. PATERSON AND U. ZWICK, *A deterministic subexponential algorithm for solving parity games*, *SIAM J. Comput.* **38** (2008), 1519–1532.
 - [42] D. KOZEN, *Results on the propositional μ -calculus*, *Theoret. Comput. Sci.* **27** (1983), 333–354.
 - [43] O. KUPFERMAN AND M. VARDI, *On bounded specifications*, LNCS volume 2250, Springer, Berlin (2001), 24–38.
 - [44] O. KUPFERMAN AND M. VARDI, $\pi_2 \cap \sigma_2 \equiv afmc$, LNCS volume 2719, Springer, Berlin (2003), 697–713.
 - [45] G. LENZI, *The μ -calculus and the hierarchy problem*, Ph.D. thesis, Scuola Normale Superiore, Pisa (1997).
 - [46] G. LENZI, *The transitive μ -calculus is Büchi definable*, *WSEAS Trans. Math.* **5** (2006), 1021–1026.
 - [47] D. LONG, A. BROWNE, E. CLARKE, S. JHA AND W. MARRERO, *An improved algorithm for the evaluation of fixpoint expressions*, LNCS volume 818, Springer, Berlin (1994), 338–350.
 - [48] R.S. LUBARSKI, *μ -definable sets of integers*, *J. Symb. Logic* **58** (1993), 291–313.

- [49] D. MARTIN, *Borel determinacy*, Ann. Math. **102** (1975), 363–371.
- [50] R. MCNAUGHTON, *Infinite games played on finite graphs*, Ann. Pure Appl. Logic **65** (1993), 149–184.
- [51] D. NIWINSKI, *Fixed point characterization of infinite behavior of finite-state systems*, Theoret. Comput. Sci. **189** (1997), 1–69.
- [52] J. OBDRZALEK, *Fast mu-calculus model checking when tree width is bounded*, LNCS volume 2725, Springer, Berlin (2003), 80–92.
- [53] J. OBDRZALEK, *Dag-width - connectivity measure for directed graphs*, in the proceedings of *ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*, ACM Press, U.S.A. (2006), 814–821.
- [54] J. OBDRZALEK, *Clique-width and parity games*, LNCS volume 4646, Springer, Berlin (2007), 54–68.
- [55] M. OTTO, *Eliminating recursion in the μ -calculus*, LNCS volume 1563, Springer, Berlin (1999), 531–540.
- [56] R. PARIKH, *Propositional game logic*, in *IEEE Proceedings on Foundations of Computer Science (FOCS 1983)*, IEEE Computer Society Press, U.S.A. (1983), 195–200.
- [57] R. PARIKH AND M. PAULY, *Game logic — An overview*, Studia Logica **75** (2003), 165–182.
- [58] V. PRATT, *Semantical considerations of Floyd-Hoare logic*, in *IEEE Proceedings on Foundations of Computer Science (FOCS 1983)*, IEEE Computer Society Press, U.S.A. (1983), 109–121.
- [59] M. RABIN, *Decidability of second order theories and automata on infinite trees*, Trans. Amer. Math. Soc. **141** (1969), 1–35.
- [60] E. ROSEN, *Modal logic over finite structures*, J. Log. Lang. Inf. **6** (1997), 427–439.
- [61] L. SANTOCANALE AND W. BELKHIR, *The variable hierarchy for the games μ -calculus*, Ann. Pure Appl. Logic **161** (2010), 690–707.
- [62] S. SCHEWE, *Solving parity games in big steps*, LNCS volume 4855, Springer, Berlin (2007), 449–460.
- [63] C. SMORYNSKI, *Self-reference and modal logic*, Springer, Berlin (1985).
- [64] C. STIRLING AND D. WALKER, *Local model checking in the modal mu-calculus*, Theoret. Comput. Sci. **89** (1991), 161–177.
- [65] R.S. STREETT AND E. A. EMERSON, *An automata theoretic decision procedure for the propositional mu-calculus*, Inform. and Comput. **81** (1989), 249–264.
- [66] B. TEN CATE AND G. FONTAINE, *An easy completeness proof for the modal μ -calculus on finite trees*, LNCS volume 6014, Springer, Berlin (2010), 161–175.
- [67] W. THOMAS, *Automata on infinite objects*, in J. VAN LEEUWEN, *Handbook of theoretical computer science. Volume B: Formal models and semantics*, North-Holland Pub. Co./MIT Press (1990), 133–192.
- [68] J. VAN BENTHEM, *Modal correspondence theory*, Ph.D. thesis, University of Amsterdam, Amsterdam (1976).
- [69] J. VAN BENTHEM, *Modal frame correspondences and fixed points*, Studia Logica **83** (2006), 133–155.
- [70] A. VISSER, *Löb’s logic meets the μ -calculus*, LNCS volume 3838, Springer Berlin (2005), 14–25.

- [71] J. VOGÉ AND M. JURDZINSKI, *A discrete strategy improvement algorithm for solving parity games*, LNCS volume 1855, Springer, Berlin (2000), 202–215.
- [72] I. WALUKIEWICZ, *Completeness of Kozen's axiomatization of the propositional μ -calculus*, Inform. and Comput. **157** (2000), 142–182.
- [73] W. ZIELONKA, *Infinite games on finitely coloured graphs with applications to automata on infinite trees*, Theoret. Comput. Sci. **200** (1998), 135–183.

Author's address:

Giacomo Lenzi
Università di Salerno
via Ponte Don Melillo, 84084 Fisciano (SA), Italy
E-mail: gilenzi@unisa.it

Received September 7, 2010
Revised October 30, 2010