From Logic to Games

Igor Walukiewicz

CNRS
LaBRI, Université Bordeaux-1
351, Cours de la Libération, 33 405, Talence
France

1 Introduction

The occasion of 25th jubilee of FSTCS gives an opportunity to look a bit further back then one normally would. In this presentation we will look at some developments in what is called formal verification. In the seventies logics occupied a principal place: Hoare logic [43], algorithmic logic [38], dynamic logic [41, 42], linear time temporal logic [55]. With a notable exception of the last one, these formalisms included programs into syntax of the logic with an idea to reduce verification to validity checking. Temporal logic was the first to advocate externalization of modeling of programs and passing from validity checking to model checking. Since the eighties, this view became predominant, and we have seen a proliferation of logical systems. We have learned that game based methods not only are very useful but also permit to abstract from irrelevant details of logical formalisms. At present games themselves take place of specification formalisms.

Roughly, model-checking can be seen as a discipline of verifying properties of labelled graphs. So, we are interested in formalisms for specifying graph properties. This formulation is misleadingly simple at the first sight. Observe, for example, that almost all the richness of first-order logic already appears over graph models, i.e., models with one binary relation. Thus, the goal is to get formalisms that are expressive and at the same time have decidable model-checking problem (and preferably with low computational complexity).

The foundations of the discipline were ready before 1980-ties. Automata theory existed already for a long time[71]. Büchi and Rabin have shown decidability of monadic second-order (MSO) theories of sequences [16] and trees [70], respectively. Martin has proven determinacy of Borel games [56]. Manna and Pnueli have already proposed a new way of looking at program verification using linear time temporal logic. Kamp's theorem gave equivalence of LTL with first-order logic over sequences [46].

Nevertheless, it is fair to say that a quarter of a century ago, at the beginning of 80-eighties, the next important period in the development of the field took place. In a relatively short interval of time a big number of significant concepts have been born. Emerson and Clarke introduced CTL and the branching/linear time distinction was clarified [25, 52]. Kozen defined the μ-calculus [48], the logic that will later bring games to the field. Independently, Büchi [17], and Gurevich and Harrington [40], arrive at understanding that the cornerstone of Rabin's

R. Ramanujam and S. Sen (Eds.): FSTTCS 2005, LNCS 3821, pp. 79–91, 2005.

decidability result for MSO theory of trees is a theorem about existence of some special strategies, now called finite memory, in some class of games. A bit later, Street and Emerson [77, 78] developed techniques to attack satisfiability problem for the μ-calculus. Around 1980 a concept of alternation was born [23]. These have given later rise to alternating automata [63] and finally to understanding that these are essentially the same as the μ-calculus.

In what follows we will give a brief introduction to the concepts described above. This will bring us in a position to discuss open problems and directions for future research. Present note is not meant to be a comprehensive survey of the discipline. Citations and results are merely chosen to demonstrate development of some lines of research, there is by far not enough place to present all important accomplishments of the field.

2 The Concepts

We need to start with presentation of some basic concepts. From 25 years perspective it is clear that they where very influential in development of the theory. The μ-calculus turned out to be important because of its purity, its expressive power and because of technical problems posed by the fixpoint operator. Old methods, like construction of syntactic models from consistent sets of formulas, are not applicable to the μ-calculus. New techniques were required, and this is where automata theory and game theory came to rescue.

2.1 The μ-Calculus

Formulas of the μ-calculus over the sets $Prop = \{p_1, p_2, \dots\}$ of *propositional constants*, $Act = \{a, b, \dots\}$ of *actions*, and $Var = \{X, Y, \dots\}$ of *variables*, are defined by the following grammar:

$$F := Prop \mid \neg Prop \mid Var \mid F \vee F \mid F \wedge F \mid \langle Act\rangle F \mid [Act]F \mid \mu\, Var.F \mid \nu\, Var.F$$

Note that we allow negations only before propositional constants. This is not a problem as we will be interested in *sentences*, i.e., formulas where all variables are bound by μ or ν. In the following, $\alpha, \beta, \dots$ will denote formulas.

Formulas are interpreted in *transition systems*, these are of the form $\mathcal{M} = \langle S, \{R_a\}_{a\in Act}, \rho\rangle$, where: S is a nonempty set of *states*, $R_a \subseteq S \times S$ is a binary relation interpreting the action a, and $\rho : Prop \to \mathcal{P}(S)$ is a function assigning to each propositional constant a set of states where this constant holds.

For a given transition system $\mathcal{M}$ and an *assignment* $V : Var \to \mathcal{P}(S)$, the set of states in which a formula α is true, denoted $\| \alpha \|_V^{\mathcal{M}}$, is defined inductively as follows:

$$\begin{aligned}
\| p \|_V^{\mathcal{M}} &= \rho(p) \qquad \| \neg p \|_V^{\mathcal{M}} = S - \rho(p) \\
\| X \|_V^{\mathcal{M}} &= V(X) \\
\| \langle a \rangle \alpha \|_V^{\mathcal{M}} &= \{ s : \exists s'. R_a(s, s') \wedge s' \in \| \alpha \|_V^{\mathcal{M}} \} \\
\| \mu X.\alpha(X) \|_V^{\mathcal{M}} &= \bigcap \{ S' \subseteq S : \| \alpha \|_{V[S'/X]}^{\mathcal{M}} \subseteq S' \} \\
\| \nu X.\alpha(X) \|_V^{\mathcal{M}} &= \bigcup \{ S' \subseteq S : S' \subseteq \| \alpha \|_{V[S'/X]}^{\mathcal{M}} \}
\end{aligned}$$

We have omitted here the obvious clauses for boolean operators and for $[a]\alpha$ formula. We will omit V in the notation if α is a sentence and will sometimes write $\mathcal{M}, s \vDash \alpha$ instead of $s \in \| \alpha \|^{\mathcal{M}}$.

The *model-checking problem for the μ-calculus* is: given a sentence α and a finite transition system $\mathcal{M}$ with a distinguished state s^0 decide if $\mathcal{M}, s^0 \vDash \alpha$.

2.2 Games

A *game* G is a tuple $\langle V_E, V_A, T \subseteq (V_E \cup V_A)^2, Acc \subseteq (V_E \cup V_A)^\omega \rangle$ where Acc is a set defining the *winning condition* and $\langle V_E \cup V_A, T \rangle$ is a graph with the vertices partitioned into those of Eve and those of Adam. We say that a vertex v' is a *successor* of a vertex v if $T(v, v')$ holds.

A *play* between Eve and Adam from some vertex $v \in V = V_E \cup V_A$ proceeds as follows: if $v \in V_E$ then Eve makes a choice of a successor, otherwise Adam chooses a successor; from this successor the same rule applies and the play goes on forever unless one of the parties cannot make a move. The player who cannot make a move looses. The result of an infinite play is an infinite path $v_0 v_1 v_2 \ldots$ This *path is winning* for Eve if it belongs to Acc. Otherwise Adam is the winner.

A *strategy* σ for Eve is a function assigning to every sequence of vertices $\boldsymbol{v}$ ending in a vertex v from V_E a vertex $\sigma(\boldsymbol{v})$ which is a successor of v. A *play respecting* σ is a sequence $v_0 v_1 \ldots$ such that $v_{i+1} = \sigma(v_i)$ for all i with $v_i \in V_E$. The *strategy σ is winning for Eve* from a vertex v iff all the plays starting in v and respecting σ are winning. A *vertex is winning for Eve* if there exists a strategy winning from it. The strategies for Adam are defined similarly. Usually we are interested in *solving games*, i.e., deciding which vertices are winninng for Eve and which for Adam.

A *strategy with memory* M is a triple:

$$c : M \times V_E \to \mathcal{P}(V), \quad up : M \times V \to M, \quad m_0 \in M$$

The role of the initial memory element m_0 and the memory update function up is to abstract some information from the sequence $\boldsymbol{v}$. This is done by iteratively applying up function:

$$up^*(m, \varepsilon) = m \quad \text{and} \quad up^*(m, \boldsymbol{v}v) = up^*(up(m, \boldsymbol{v}), v)$$

This way, each sequence $\boldsymbol{v}$ of vertices is assigned a memory element $up^*(m_0, \boldsymbol{v})$. Then the choice function c defines a strategy by $\sigma(\boldsymbol{v}v) = c(up^*(m_0, \boldsymbol{v}), v)$. The

strategy is *memoryless* iff $\sigma(\boldsymbol{v}) = \sigma(\boldsymbol{w})$ whenever $\boldsymbol{v}$ and $\boldsymbol{w}$ end in the same vertex; this is a strategy with a memory M that is a singleton.

In most of the cases here the winning conditions $Acc \subseteq V^\omega$ will be *Muller conditions*: that is, there will be a colouring $\lambda : V \to Colours$ of the set of vertices with a finite set of colours and a set $\mathcal{F} \subseteq \mathcal{P}(Colours)$ that define the winning sequences by:

$$\boldsymbol{v} \in Acc \quad \text{iff} \quad \mathrm{Inf}_\lambda(\boldsymbol{v}) \in \mathcal{F}$$

where $\mathrm{Inf}_\lambda(\boldsymbol{v})$ is the set of colours appearing infinitely often on $\boldsymbol{v}$.

An important special case is a *parity condition*. It is a condition determined by a function $\Omega : V \to \{0, \ldots, d\}$ in the following way:

$$Acc = \{v_0 v_1 \ldots \in V^\omega : \limsup_{i \to \infty} \Omega(v_i) \text{ is even}\}$$

Hence, in this case, the colours are natural numbers and we require that the biggest among those appearing infinitely often is even. This condition was discovered by Mostowski [60] and is the most useful form of Muller conditions. It is the only Muller condition that guarantees existence of memoryless strategies [33, 61, 58]. It is closed by negation (the negation of a parity condition is a parity condition). It is universal in the sense that very game with a Muller condition can be reduced to a game with a parity condition [60].

2.3 Between Games and Formulas

The truth of a given formula in a given model can be characterized by games. To see this, consider the task of checking if a propositional formula (in a positive normal form) is true in a given valuation. If the formula is a disjunction then Eve should choose one of the disjuncts that she believes is true; if it is a conjunction then Adam should choose one of the conjuncts he believes is false. The game continues until it arrives at a literal (a proposition or its negation). Eve wins iff the literal is true in the valuation fixed at the beginning. It is easy to see that Eve has a winning strategy in this game iff the initial formula is true in the valuation.

Observe that we can define a similar game for almost any logic, just using directly the clauses defining its semantics. For example, for first-order logic Eve would choose in the case of disjunction and existential quantifier, and Adam in the case of conjunction and universal quantifier. This view is of course well known. The reason why it is not used too much in the context of first-order logic is that the game becomes quite complicated. One can consider Ehrenfeucht-Fraïsé games as a way of hiding these complications at the cost of limiting the scope of applicability of the concept.

While it is clear how to define game rules for disjunction, conjunction, and most other cases, it is much less clear what to do with fixpoints. The best we can do when we want to see if a formula $\mu X.\alpha(X)$ holds is to check if its unwinding $\alpha(\mu X.\alpha(X))$ holds. Such an unwinding rule introduces potential of infinite plays

as for $\mu X.X$. The other problem is that for the greatest fixpoint $\nu X.\alpha(X)$ we cannot do better but suggest the same rule. One of the most important developments in these 25 years is to admit infinite plays and to realize that fixpoints give rise to a parity condition on infinite plays: least fixpoints are given odd ranks, greatest fixpoints even ranks, and the exact value of the rank depends on the nesting depth (see [76] for details).

Summarizing, one can look at the formula as a kind of schema that when put together with a model defines a game. Observe that a schema by itself does not define a game directly; putting it differently, the satisfiability question requires more than just examining the structure of the formula. We see the same phenomenon in a formalism of alternating automata. It is a very beautiful fact that the to formalisms agree. Actually it is one of the cornerstones of the whole theory.

2.4 Alternating Automata

An *alternating automaton* on on transition systems is a tuple:

$$\mathcal{A} = \langle A, P, Q^{\exists}, Q^{\forall}, q^0, \delta : Q \times \mathcal{P}(P) \to \mathcal{P}(A \times Q), Acc \rangle$$

where $A \subseteq Act \cup \{id\}$, $P \subseteq Prop$ are finite set of actions and propositions, respectively, relevant to the automaton. Set Q is a finite set of states partitioned into existential, $Q^{\exists}$, and universal, $Q^{\forall}$ states. State $q^0 \in Q$ is the initial state of the automaton and δ is the transition function that assigns to each state and label, which is valuation relevant propositions, a set of possible moves. An intuitive meaning of a move $(a, q') \in A \times Q$ is to move over an edge labelled a and change the state to q'. The action id is a self-loop permitting to stay in the same node. Finally, $Acc \subseteq Q^{\omega}$ is an acceptance condition.

The simplest way to formalize the notions of a run and of an acceptance of an automaton is in terms of games. Given an automaton $\mathcal{A}$ as above and a transition system $\mathcal{M} = \langle S, \{R_a\}_{a \in Act}, \rho \rangle$ we define the *acceptance game* $G(\mathcal{A}, P) = \langle V_E, V_A, T, Acc_G \rangle$ as follows:

- The set of vertices for Eve is $(Q^{\exists} \times S)$.
- The set of vertices for Adam is $(Q^{\forall} \times S)$.
- From each vertex (q, s), for every $(a, q') \in \delta(q, \lambda(s))$ and $(s, s') \in R_a$ we have an edge in T to (q', s'); we assume that R_{id} is the identity relation on states.
- The winning condition Acc_G consists of the sequences:

  $$(q_0, s_0)(q_1, s_1) \dots$$

  such that the sequence $q_0, q_1 \dots$ is in Acc, i.e., it belongs to the acceptance condition of the automaton.

Let us see how to construct an automaton equivalent to a sentence α of the μ-calculus (we do not admit free variables in α). The states of the automaton $\mathcal{A}_\alpha$ will be the subformulas of the formula α plus two states $\top$ and $\bot$. The initial state will be α. The action and proposition alphabets of $\mathcal{A}_\alpha$ will consist of the actions and propositions that appear in α. The transitions will be defined by:

- $\delta(p, v) = \top$ if $p \in v$ and $\bot$ otherwise;
- $\delta(\beta_1 \vee \beta_2, v) = \delta(\beta_1 \wedge \beta_2, v) = \{(id, \beta_1), (id, \beta_2)\}$;
- $\delta(\langle a \rangle \beta, v) = \delta([a]\beta, v) = \{(a, \beta)\}$;
- $\delta(\mu X.\beta(X), v) = \delta(\nu X.\beta(X), v) = \{(id, \beta(X))\}$;
- $\delta(X, v) = \{(id, \beta(X))\}$.

The symbols in the last rule demand some explications. Here X is a variable and $\beta(X)$ is the formula to which it is bound, i.e., we have $\mu X.\beta(X)$ or $\nu X.\beta(X)$ in α. We can suppose that X is bound precisely once in α as we can always rename bound variables.

Observe that the rules for conjunction and disjunction are the same. The difference is that a disjunction subformula will be an existential state of $\mathcal{A}_\alpha$ and the conjunction subformula an universal one. Similarly for $\bot$, $\top$ as well as for $\langle a \rangle$ and $[a]$ modalities. This means, in particular, that $\top$ is an accepting state as there are no transitions from $\top$ and Adam looses immediately in any position of the form $(\top, s)$.

It remains to define the acceptance condition of $\mathcal{A}_\alpha$. It will be the parity condition where all the subformulas but variables have rank 0. To define the rank of a variable X we look at the formula it is bound to. If it is $\mu X.\beta(X)$ then the rank of X is $2d + 1$ where d is the nesting depth of $\mu X.\beta(X)$. If it is is $\nu X.\beta(X)$ then it is $2d$. For the precise definition of the nesting depth we refer the reader to [4], here it suffices to say that the principle is the same as for quantifier depth in first-order logic.

We will not discuss here, not too difficult, proof that this construction gives indeed an automaton equivalent to the formula. What is worth pointing out is that the translation in the other direction is also possible. From a given alternating automaton one can construct an equivalent formula of the μ-calculus. This equivalence is a very good example of a correspondence between formula and diagram based formalisms as advocated by Wolfgang Thomas [79]. The μ-calculus is compositional, it permits doing proofs by induction on the syntax. Automata are better for algorithmic issues and problems such as minimization.

The last remark we want to make here is about satisfiability. The above reduction shows that the satisfiability question for the μ-calculus can be solved via emptiness problem for alternating automata. This in turn requires transformation of alternating to nondeterministic automata or in other words, elimination of universal branching. When we look back we can see that this is an universal phenomenon that appears even in the case of propositional logic.

3 Perspectives

One of the obvious problems that resisted over the last 25 years is the model checking problem for the μ-calculus. Equivalently, it is the problem of solving parity games. In this formulation it is a, potentially simpler, instance of the problem of solving stochastic games [44] which complexity is open for quite some time. There are at least two directions of research that are connected to this problem and that are also interesting in their own right.

One direction is to find polynomial-time algorithms for some restricted classes of games. For example, for any constant, games whose graphs have tree-width bounded by this constant can be solved in polynomial time [68]. Recently, a new graph complexity measure, called entanglement, has been proposed and the same result for graphs of bounded entanglement has been proved [8]. In the future it would be interesting to consider the case of clique-width. Tree-width is connected to MSO logic where quantification over transitions is permitted. It is known that in this logic each μ-calculus formula is equivalent to a formula of quantifier depth 3. Clique-width [27] is linked to MSO logic where only quantification over states is permitted. In this case it is open whether a finite number of quantifier alternations suffices to capture the whole μ-calculus.

The other direction is to consider the model-checking and game solving problems for graphs represented implicitly. A simple example is a graph represented as a synchronized product of transition systems. In this case even alternating reachability is EXPTIME-complete (see [30] for more detailed analysis) but the model-checking problem for the whole μ-calculus stays also in EXPTIME. The other possibility is to consider configuration graphs of some type of machines. In recent years pushdown graphs, i.e., graphs of configurations of pushdown machines have attracted considerable attention [62, 9, 34, 50, 74, 83]. One research direction is to find interesting and decidable classes of properties of pushdown systems [11, 36, 75]. The other direction is to go forward to more complicated cases like higher order pushdowns [18, 19], higher order recursive program schemes [45, 47] or pushdowns with parallel composition [10]. The biggest challenge here is to push the decidability frontier.

The understanding that the μ-calculus corresponds exactly to games with parity conditions suggest to look for other winning conditions with interesting properties. For finite Muller conditions we know how to calculate a memory required to win [32]. Recently, more general winning conditions were investigated as for example Muller conditions over infinite number of colours [39]. A particular case of such a condition is when colours are natural numbers and the winner is decided by looking at the parity of the smallest number appearing infinitely often (additionally we can assume that Eve wins if there is no such number). It turns out that this infinite kind of a parity condition is the only type of infinite Muller condition that guarantees the existence of memoryless strategies in all games. It is important to add that for this result to hold all positions of the game need to have a colour assigned. If we permit partial assignments of colours or put coloring on edges of the game graph and not on positions then only ordinary (i.e. finite) parity conditions admit memoryless strategies [26]. In the recent paper [37] this later result is extended to include also quantitative conditions such as mean or discounted pay-off.

While we know already a great deal about the μ-calculus itself [4], there still remains a lot to explore. One of the obvious research topics suggested by the syntax of the logic is that of the alternation hierarchy of fixpoint operators. Curiously, as the translation presented above shows, the alternation depth of the formula corresponds to the size of a parity condition in the equivalent al-

ternating automaton. Thus, one can equivalently study the later hierarchy. The infiniteness of the hierarchy for the μ-calculus was shown by Bradfield [14] and for alternating tree automata independently by Bradfield [15] and Arnold [3]. It is worth noting that hierarchy questions for nondeterministic tree automata were solved ten years earlier by Niwiński [64], and for the even simpler case of deterministic automata, another ten years back by Wagner [82]. (As a side remark let us mention that quite recently Arnold and Santocanale has shown a surprising behaviour of diagonal classes [5].) Once the basic hierarchy questions are resolved, the next challenge is to provide algorithms for determining the level in the hierarchy of a given recognizable language. The first step was to give a polynomial time algorithm for computing the level in the hierarchy of deterministic automata [65]. Next, Urbański [80] has shown that it is decidable if a deterministic Rabin tree automaton is equivalent to a nondeterministic Büchi one. Actually, the problem is also in PTIME [66]. More recently [67], the case of deterministic tree automata was completely solved. There are forbidden pattern characterizations for all the levels of the hierarchy of nondeterministic automata; that is, given a deterministic automaton one can tell by examining its structure to which level of the hierarchy of nondeterministic automata it belongs to. This also solves the problem for levels of alternating automata hierarchy as all deterministic languages are recognizable by co-Büchi alternating automata. The challenge for the future is to calculate hierarchy levels for nondeterministic automata.

There are numerous other directions of active research. We will describe just three more very briefly here, referring the reader to the cited papers for details.

Games as well as logics and automata can be augmented with real-time. While real-time automata are around for some time now [2], there is no standard logic for real-time. This is partially due to the fact that timed-automata are not closed under complement and it is difficult to decide on some other good class of real-time properties. Also quantitative reachability problems, like minimizing reachability cost, appear to be interesting [1, 12, 21].

The rules of playing games may be extended [28]: one may allow concurrent moves when two players choose moves independently and the game proceeds to the state that is a function of the two choices. An example of "paper, scissors, stone" game shows that randomized strategies are sometimes necessary to win in such games. This means that now a player does not win for sure, but only with certain probability; the maximal such probability is called the value of the player. Another extension is to allow randomized positions where a successor is chosen randomly with respect to some probability distribution. The quantitative determinacy result of Martin [57] states that in every game with concurrent moves and randomized positions the values for Eva and Adam sum up to 1. In [28] de Alfaro and Majumdar show how to calculate the values of a game using appropriate extension of the μ-calculus. It can also happen that the objectives of the two players are not antagonistic, in this case we talk about Nash equilibria rather than values of games. Recently [24], Chatterjee has shown how to calculate

Nash equilibria for a very general class of concurrent, stochastic, nonzero-sum, infinite games.

Finally, each of these game models can be applied to synthesis [69, 72, 49, 20]. The synthesis problem is to construct a system from a given specification. It is often solved by reduction to the problem of finding a strategy in some game [6]. If the problem mentions real-time then the game will have real-time constraints [7, 29, 31, 13, 22]. If the problem concerns distributed setting then either the reduction or the game model will have to take it into account [73, 53, 54, 51, 59, 35]. The number of choices is truly overwhelming and we need to understand much better in what cases synthesis is feasible.

References

[1] R. Alur, M. Bernadsky, and P. Madhusudan. Optimal reachability for weighted timed games. In *ICALP*, volume 3124 of *Lecture Notes in Computer Science*, pages 122–133, 2004.

[2] R. Alur and P. Madhusudan. Decision problems for timed automata: A survey. In *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *Lecture Notes in Computer Science*, pages 1–24, 2004.

[3] A. Arnold. The mu-calculus alternation-depth hierarchy is strict on binary trees. *RAIRO–Theoretical Informatics and Applications*, 33:329–339, 1999.

[4] A. Arnold and D. Niwiski. *The Rudiments of the Mu-Calculus*, volume 146 of *Studies in Logic*. North-Holand, 2001.

[5] A. Arnold and L. Santocanale. Ambiguous classes in the games mu-calculus hierarchy. In *FOSSACS 03*, volume 2620 of *Lecture Notes in Computer Science*, pages 70–86, 2003.

[6] A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303(1):7–34, 2003.

[7] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Contrloller synthesis for timed automata. In *Proc. IFAC Symp. System Structure and Control*, pages 469–474, 1998.

[8] D. Berwanger and E. Grädel. Entanglement - a measure for the complexity of directed graphs with applications to logic and games. In *LPAR 2004*, volume 3452 of *Lecture Notes in Computer Science*, pages 209–223, 2004.

[9] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Applications to model checking. In *CONCUR'97*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150, 1997.

[10] A. Bouajjani, M. Mueller-Olm, and T. Touili. Regular symbolic analysis of dynamic networks of pushdown systems. In *CONCUR'05*, volume 3653 of *Lecture Notes in Computer Science*, 2005.

[11] A. Bouquet, O. Serre, and I. Walukiewicz. Pushdown games with the unboundedness and regular conditions. In *FSTTCS'03*, volume 2914 of *Lecture Notes in Computer Science*, pages 88–99, 2003.

[12] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *FSTTCS*, Lecture Notes in Computer Science, 2004.

[13] P. Bouyer, D. D'Souza, P. Madhusudan, and A. Petit. Timed control with partial observability. In *CAV'03*, volume 2725 of *Lecture Notes in Computer Science*, pages 180–192, 2003.

[14] J. Bradfield. The modal mu-calculus alternation hierarchy is strict. *Theoretical Computer Science*, 195:133–153, 1997.
[15] J. Bradfield. Fixpoint alternation: Arithmetic, transition systems, and the binary tree. *RAIRO–Theoretical Informatics and Applications*, 33:341–356, 1999.
[16] J. R. Büchi. On the decision method in restricted second-order arithmetic. In *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford Univ. Press, 1960.
[17] J. R. Buchi. State strategies for games in $F_{\sigma\delta} \cap G_{\delta\sigma}$. *Journal of Symbolic Logic*, 48:1171–1198, 1983.
[18] T. Cachat. Symbolic strategy synthesis for games on pushdown graphs. In *ICALP'02*, volume 2380 of *Lecture Notes in Computer Science*, pages 704–715, 2002.
[19] T. Cachat. Uniform solution of parity games on prefix-recognizable graphs. In A. Kucera and R. Mayr, editors, *Proceedings of the 4th International Workshop on Verification of Infinite-State Systems*, volume 68 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2002.
[20] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.
[21] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR'05*, Lecture Notes in Computer Science, 2005.
[22] F. Cassez, T. Henzinger, and J. Raskin. A comparison of control problems for timed and hybrid systems. In *Hybrid Systems Computation and Control (HSCC'02)*, number 2289 in Lecture Notes in Computer Science, pages 134–148, 2002.
[23] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the Association of Computing Machinery*, 28(1):114–133, 1981.
[24] K. Chatterjee. Two-player nonzero-sum omega-regular games. In *CONCUR'05*, Lecture Notes in Computer Science, 2005.
[25] E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981.
[26] T. Colcombet and D. Niwiński. On the positional determinacy of edge–labeled games. Submitted, 2004.
[27] B. Courcelle and P. Weil. The recognizability of sets of graphs is a robust property. To appear in Theoretical Computer Science, http://www.labri.fr/Perso/~weil/publications/.
[28] L. de Alfaro. Quantitative verification and control via the mu-calculus. In *CONCUR'03*, volume 2761 of *Lecture Notes in Computer Science*, pages 102–126, 2003.
[29] L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR'03*, volume 2761 of *Lecture Notes in Computer Science*, pages 142–156, 2003.
[30] S. Demri, F. Laroussinie, and P. Schnoebelen. A parametric analysis of the state exposion problem in model checking. In *STACS'02*, volume 2285 of *Lecture Notes in Computer Science*, pages 620–631, 2002.
[31] D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *STACS'02*, volume 2285 of *Lecture Notes in Computer Science*, pages 571–582, 2002.
[32] S. Dziembowski, M. Jurdzinski, and I. Walukiewicz. How much memory is needed to win infinite games. In *LICS*, pages 99–110, 1997.

[33] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. FOCS'91*, pages 368–377, 1991.
[34] J. Esparza and A. Podelski. Efficient algorithms for pre star and post star on interprocedural parallel flow graphs. In *POPL'00: Principles of Programming Languages*, 2000.
[35] B. Finkbeiner and S. Schewe. Uniform distributed synthesis. In *LICS05*, 2005.
[36] H. Gimbert. Parity and explosion games on context-free graphs. In *CSL'04*, volume 3210 of *Lecture Notes in Computer Science*, pages 56–70, 2004.
[37] H. Gimbert and W. Zielonka. When can you play positionally? In *MFCS'04*, volume 3153 of *Lecture Notes in Computer Science*, 2004.
[38] G.Mirkowska and A.Salwicki. *Algorithmic Logic*. D.Reidel PWN, 1987.
[39] E. Grädel and I. Walukiewicz. Positional determinacy of infnite games, 2004. Submitted.
[40] Y. Gurevich and L. Harrington. Trees, automata and games. In *14th ACM Symp. on Theory of Computations*, pages 60–65, 1982.
[41] D. Harel. Dynamic logic. In *Handbook of Philosophical Logic Vol II*, pages 497–604. D.Reidel Publishing Company, 1984.
[42] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
[43] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12:576–585, 1969.
[44] A. Hoffman and R. Karp. On nonterminating stochastic games. *Management Science*, 12:369–370, 1966.
[45] C.-H. L. O. K. Aehlig, J. G. de Miranda. The monadic second order theory of trees given by arbitrary level-two recursion schemes is decidable. In *TLCA'05*, volume 3461 of *Lecture Notes in Computer Science*, pages 39–54, 2005.
[46] H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, 1968.
[47] T. Knapik, D. Niwinski, P. Urzyczyn, and I. Walukiewicz. Unsafe grammars and panic automata. In *ICALP'05*, volume 3580 of *Lecture Notes in Computer Science*, pages 1450–1461, 2005.
[48] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
[49] R. Kumar and V. K. Garg. *Modeling and control of logical discrete event systems*. Kluwer Academic Pub., 1995.
[50] O. Kupferman and M. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In *Proceedings of CAV'00*, volume 1855 of *Lecture Notes in Computer Science*, pages 36–52. Springer Verlag, 2000.
[51] O. Kupferman and M. Vardi. Synthesizing distributed systems. In *Proc. 16th IEEE Symp. on Logic in Computer Science*, 2001.
[52] L. Lamport. "sometime" is sometimes "not never" – on the temporal logic of programs. In *POPL'80*, pages 174–185, 1980.
[53] P. Madhusudan. *Control and Synthesis of Open Reactive Systems*. PhD thesis, University of Madras, 2001.
[54] P. Madhusudan and P. Thiagarajan. A decidable class of asynchronous distributed controllers. In *CONCUR'02*, volume 2421 of *Lecture Notes in Computer Science*, 2002.
[55] Z. Manna and A. Pnueli. Verification of the concurrent programs: the temporal framework. In R.Boyer and J.Moore, editors, *The Correctness Problem in Computer Scince*, pages 215–273. Academic Press, 1981.
[56] D. Martin. Borel determinacy. *Ann. Math.*, 102:363–371, 1975.

[57] D. Martin. The determinacy of Blackwell games. *The Journal of Symbolic Logic*, 63(4):1565–1581, 1998.
[58] R. McNaughton. Infinite games played on finite graphs. *Ann. Pure and Applied Logic*, 65:149–184, 1993.
[59] S. Mohalik and I. Walukiewicz. Distributed games. In *FSTTCS'03*, volume 2914 of *Lecture Notes in Computer Science*, pages 338–351, 2003.
[60] A. W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Fifth Symposium on Computation Theory*, volume 208 of *LNCS*, pages 157–168, 1984.
[61] A. W. Mostowski. Games with forbidden positions. Technical Report 78, University of Gdansk, 1991.
[62] D. Muller and P. Schupp. The theory of ends, pushdown automata and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985.
[63] D. Muller and P. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.
[64] D. Niwiński. On fixed-point clones. In *Proc. 13th ICALP*, volume 226 of *LNCS*, pages 464–473, 1986.
[65] D. Niwiński and I. Walukiewicz. Relating hierarchies of word and tree automata. In *STACS'98*, volume 1373 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.
[66] D. Niwiński and I. Walukiewicz. A gap property of deterministic tree languages. *Theoretical Computer Science*, 303(1):215–231, 2003.
[67] D. Niwiński and I. Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Electr. Notes Theor. Comput. Sci.*, 123:195–208, 2005.
[68] J. Obdrzalek. Fast mu-calculus model checking when tree-width is bounded. In *CAV'03*, volume 2725 of *Lecture Notes in Computer Science*, pages 80–92, 2003.
[69] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. ACM POPL*, pages 179–190, 1989.
[70] M. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.
[71] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, pages 114–125, 1959. Reprinted in Sequential machines (editor E. F. Moore), Addison-Wesley, Reading, Massachusetts, 1964, pages 63-91.
[72] P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(2):81–98, 1989.
[73] K. Rudie and W. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Trans. on Automat. Control*, 37(11):1692–1708, 1992.
[74] O. Serre. Note on winning positions on pushdown games with ω-regular conditions. *Information Processing Letters*, 85:285–291, 2003.
[75] O. Serre. Games with winning conditions of high Borel complexity. In *ICALP'04*, volume 3142 of *Lecture Notes in Computer Science*, pages 1150–1162, 2004.
[76] C. Stirling. *Modal and Temporal Properties of Processes*. Texts in Computer Science. Springer, 2001.
[77] R. S. Streett and E. A. Emerson. The propositional mu-calculus is elementary. In *ICALP*, volume 172 of *Lecture Notes in Computer Science*, pages 465–472, 1984.
[78] R. S. Streett and E. A. Emerson. An automata theoretic procedure for the propositional mu-calculus. *Information and Computation*, 81:249–264, 1989.
[79] W. Thomas. Logic for computer science: The engineering challenge. volume 2000 of *Lecture Notes in Computer Science*, pages 257–267, 2002.

[80] T. Urbański. On deciding if deterministic Rabin language is in Büchi class. In *ICALP'00*, volume 1853 of *Lecture Notes in Computer Science*, pages 663–674, 2000.

[81] M. Y. Vardi and P.Wolper. Automata theoretic techniques for modal logics of programs. In *Sixteenth ACM Symposium on the Theoretical Computer Science*, 1984.

[82] K. Wagner. Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen. *J. Inf. Process. Cybern. EIK*, 13:473–487, 1977.

[83] I. Walukiewicz. Pushdown processes: Games and model checking. *Information and Computation*, 164(2):234–263, 2001.