Symbolic Model Checking for μ-calculus Requires Exponential Time

A. Rabinovich
Department of Computer Science
Tel Aviv University, Israel 69978
e.mail: rabino@math.tau.ac.il

1 Introduction

Model checking [3] is a very popular paradigm for automatic verification of properties of finite state systems like those defined by circuits or communication protocols.

A system is interpreted as a finite state labeled transition system or equivalently as a finite Kripke structure. If *sys* contains k concurrent components, each with m states, then the Kripke structure described by *sys* may have m^k states. Hence, the size of the Kripke structure might be exponential in the size $m \times k$ of its description. This phenomenon is known as the state explosion problem. To avoid the state explosion, a method called symbolic model checking was proposed in [12]. This method avoids building a state graph by using propositional formulas to represent sets and relations.

Required properties of a system are formulated by formulas in temporal or modal formalisms like LTL, CTL, CTL* or μ-calculus (see surveys [4, 14]). The μ-calculus [10] is a powerful language for expressing properties of Kripke structures by using the least fixed point operator. It provides a single, uniform and elegant framework subsuming most temporal and modal logics of programs [5].

In this paper we investigate the complexity of the following decision problem.
Symbolic Model Checking for μ-calculus:
Input: A description of a finite state Kripke structure K and its state s by propositional formulas, and a μ-calculus formula ϕ.
Question: Determine whether the state s of the Kripke structure K satisfies the property defined by ϕ.

Notice that the size of an instance of the model checking problem is the size of the description of K, s, plus the size of ϕ. This is a natural measurement. Indeed, in order to verify a circuit with 1000 Flip-Flops, a model checker will be provided with the graph of the circuit rather than with its state transition diagram (Kripke structure) that might contain 2^{1000} states.

A straightforward algorithm for symbolic model checking will construct a Kripke structure K from its description and then evaluate the formula ϕ in K. A naive algorithm to evaluate a μ-formula in K may require $O(n^c)$ iterations, where c is the depth of nesting of fix-points and n is the number of states in K (see [5] for more efficient algorithms). The complexity of this naive algorithm is $O(|K|^{|\phi|})$. Since the size of K is exponential in the size of its description d, the time complexity of this algorithm is $O(2^{|d|\times|\phi|})$. Hence, even for the fixed formula ϕ this algorithm is exponential. We will show

Theorem 1 *There is $c > 1$ such that the symbolic model checking problem for μ-calculus cannot be solved in deterministic time $c^{n/log^2 n}$.*

This lower bound is proved by a reduction from the membership problem for linear space alternating Turing machines. The key fact we rely on is

Theorem 2 *(Chandra, Kozen, Stockmeyer [2]) ASPACE($s(n)$) $= \bigcup_{c>1}$DTIME($c^{s(n)}$).*

We will define a μ-formula $ACCEPT$ which specifies the acceptance conditions for alternating Turing machines (see Proposition 5). It is not difficult to show that the computation of a linear space alternating Turing machine on an input of size n can be encoded by propositional formulas of size $O(nlog^2n)$ (see Proposition 6). Hence, the membership problem for an alternating Turing machine operating in linear space is reducible to the model checking problem of size $O(nlog^2n)$. Therefore, Theorem 1 follows immediately from Theorem 2, and Propositions 5 and 6.Actually, our proof shows

Theorem 3 *For a fixed μ-calculus formula $ACCEPT$ there is no algorithm that runs in time $O(c^{n/log^2 n})$ and checks whether a state of the Kripke structure described by a propositional formula of size n satisfies $ACCEPT$.*

We conjecture that the lower bound can be improved to $O(c^{n/logn})$. We do not know whether the lower bound $O(c^n)$ can be shown thus closing the gap between the upper and the lower bound for checking a fixed μ-formula.

2 Preliminaries

2.1 Kripke Structures

Let $\{p_1, \dots, p_n\}$ be a finite set of symbols. A Kripke structure for the signature $\{p_1, \dots, p_n\}$ is a tuple $K = (S,\ R,\ \ P_1, \dots, P_n)$, where S is a set of states, R is a binary relation on S and $P_1, \dots P_n$ are subsets of S. A finite Kripke structure is a Kripke structure with a finite number of states.

Finite Kripke structures can be described by propositional formulas. Let $State(x_1, \dots, x_k)$, $\phi_1(x_1, \dots, x_k), \dots, \phi_1(x_1, \dots, x_k)$, $Next(x_1, \dots, x_k,\ x'_1, \dots, x'_k)$ be propositional formulas, where $x_1, \dots, x_k,\ x'_1, \dots x'_k$ are distinct propositional variables. The Kripke structure $K = (S,\ R,\ \ P_1, \ldots, P_n)$ described by these formulas is defined as follows:

- S is the set of truth assignments to $\{x_1, \ldots, x_k\}$ that satisfy $State(x_1, \ldots, x_k)$.
- P_i is the set of truth assignments to $\{x_1, \ldots, x_k\}$ that satisfy $State(x_1, \ldots, x_k) \wedge \phi_i(x_1, \ldots, x_k)$ (for $i = 1, \ldots n$).
- $R(\rho,\ \rho')$ holds if ρ (respectively ρ') is the truth assignment to $\{x_1, \ldots, x_k\}$ (respectively to $\{x'_1, \ldots, x'_k\}$) that satisfies $State(x_1, \ldots, x_k)$ (respectively $State(x'_1, \ldots, x'_k)$), and the truth assignment ρ'' that coincides with ρ on $\{x_1, \ldots, x_k\}$ and with ρ' on $\{x'_1, \ldots, x'_k\}$ satisfies $Next$.

A formula $\phi(x_1, \ldots x_n)$ will describe the subsets of the states of K that satisfy ϕ, i.e., those truth assignments that satisfy $\phi \wedge State$.

Observe that the size of a Kripke structure might be exponential in the size of propositional formulas that describe it.

2.2 Propositional μ-calculus

The formulas of the propositional μ-calculus are defined by the following grammar:

$$F ::= X \ |p\ |F \wedge F\ |\neg F\ |\Box F\ |\Diamond F\ |\mu X.F,$$

where X and p range over variable and and over propositional symbols respectively. It is required that the variables bound by the fixed point operator μ must be in the scope of an even number of negations. A μ-calculus formula that contains propositional symbols $\{p_1, \ldots, p_k\}$ is interpreted in the Kripke structures with the signature $\{p_1, \ldots, p_k\}$. We will write $[\![\phi]\!]_K e$ for the set of states defined by ϕ in a structure $K = (S,\ R,\ \ P_1, \ldots, P_n)$ and an environment $e :\ Var \rightarrow 2^S$.

The definition of $[\![\phi]\!]_K e$ is provided below. For a closed formula ϕ, a Kripke structure K and its state s we write $(K,\ s) \models \phi$ for $s \in [\![\phi]\!]_K$.

1. $[\![p_i]\!]_K e \triangleq P_i$
2. $[\![X_i]\!]_K e \triangleq e(X_i)$
3. $[\![F_1 \wedge F_2]\!]_K e \triangleq [\![F_1]\!]_K e \cap [\![F_2]\!]_K e$
4. $[\![\neg F]\!]_K e \triangleq S - [\![F]\!]_K e$
5. $[\![\Box F]\!]_K e \triangleq \{s\ |\forall t.\ R(s,t) \text{ implies } t \in [\![F]\!]_K e\}$
6. $[\![\Diamond F]\!]_K e \triangleq \{s\ |\exists t.\ R(s,t) \text{ and } t \in [\![F]\!]_K e\}$
7. $[\![\mu X.\ F]\!]_K e$ is the least fixed point of the function $\lambda S'.\ [\![F]\!]_K e[X \rightarrow S']$, where $e[X \rightarrow S']$ is the environment which is the same as e except that it has the value S on X.

2.3 Alternating Turing Machine

An alternating Turing machine [2] is a useful technical tool in the proofs of exponential lower bounds. The definition of an alternating Turing machine is similar to that of nondeterministic Turing machine except that some states are declared to be universal and others are declared to be existential. This partition of states is used in the definition of the acceptance condition for an alternating Turing machine. In this subsection we just repeat the definition of an alternating Turing machine from [7].

A one-tape alternating Turing machine is a tuple $M = (E,\ U,\ q_0,\ \Delta, \Gamma, b,\ \nu)$, where E is the set of existential states, U is the set of universal states, $q_0 \in U$ is the initial state, Δ is the input alphabet, Γ is the output alphabet, $b \in \Gamma - \Delta$ is the blank symbol and $\nu \subset (E \times U \cup U \times E) \times \Gamma \times \Gamma \times \{L,\ R\}$ is the next move relation. The set $E \cup U$ of all states is denoted by Q.

A configuration is a member of $\Gamma^* Q \Gamma^+$ and represents a complete state of the Turing machine. A universal configuration is a member of $\Gamma^* U \Gamma^+$, while an existential configuration is a member of $\Gamma^* E \Gamma^+$.

Let $\alpha = xq\sigma y$ be a configuration, where $\sigma \in \Gamma,\ x, y \in \Gamma^*$ and $q \in Q$. We define $tape(\alpha) = x\sigma y$, and $state(\alpha) = q$. Let $\beta = x'q'\sigma'y'$ be a configuration, where $\sigma' \in \Gamma,\ x', y' \in \Gamma^*$ and $q' \in Q$. We say that β is a next configuration of α (notations $\alpha \to \beta$) if for some $\tau \in \Gamma$, either

$$(q,\ q',\ \sigma,\ \tau,\ L) \in \nu,\ x'\sigma' = x \text{ and } y' = \tau y,$$

or

$$(q,\ q',\ \sigma,\ \tau,\ R) \in \nu, \text{ and } \ x' = x\tau \text{ and } \sigma'y' = \tau y \text{ or } (y = y' = \epsilon \text{ and } \sigma' = b).$$

A *computation sequence* is a sequence of configurations $\alpha_1 \ldots \alpha_k$ for which $\alpha_i \to \alpha_{i+1}$, $1 \le i < k$. A configuration β is *reachable* from a configuration α if there exists a computation sequence $\alpha_1 \ldots \alpha_k$ with $\alpha = \alpha_1$ and $\beta = \alpha_n$.

Acceptance conditions. A trace of M is a set C of pairs $(\alpha,\ t)$, where α is a configuration and t is a natural number, such that

1. If $(\alpha,\ t) \in C$ and α is a universal configuration, then for every next configuration β of α, there is a $t' < t$ for which $(\beta,\ t') \in C$ and
2. If $(\alpha,\ t) \in C$ and α is an existential configuration, then there exists a next configuration β of α and a $t' < t$ for which $(\beta,\ t') \in C$.

A configuration α is an accepting configuration if there exists a natural number t and a trace C such that $(\alpha,\ t) \in C$. The set accepted by M is

$$L(M) = \{x \in \Delta^* \ :\ \text{there exists } t \in Nat \text{ and a trace } C \text{ of } M \text{ such that } (q_0 x,\ t) \in C\}.$$

A trace C uses space at most s if for every $(\alpha,\ t) \in C$, the length of α is at most $s+1$. An alternating Turing machine M operates in space $s(n)$ if for every x of length n the length of any configuration β reachable from the configuration q_0x is at most $s(n)+1$. $ASPACE(s(n))$ is the class of sets accepted by alternating Turing machines which operates in space $s(n)$.

Remark 4 The above definition only slightly differs from the definition in [7]. Namely, we require that the initial state is a universal state and that ν alternates between the universal and the existential states. For every alternating Turing machine that violates the above requirement it is easy to construct an equivalent (i.e., accepting the same set of strings) alternating Turing machine that satisfies these requirements and operates in the same space.

3 Reduction

First we show that the acceptance conditions of an alternating Turing machine can be described by a μ-formula. Moreover, the formula is independent of a machine (see Proposition 5). Then, the computations of a linear space alternating Turing machine will be succinctly described by propositional formulas (see Proposition 6).

Let M be an alternating Turing machine. We define K_M to be the Kripke structure (Conf, $\to$, Uconf), where Conf is the set of configurations of M, and $\to$ is the next configuration relation (i.e., $\alpha \to \beta$ if β is a next configuration of α), and Uconf is the set of the universal configurations of M. We define K^n_M to be the Kripke structure (Conf_n, $\to_n$, Uconf_n), where Conf_n (respectively Uconf_n) is the set of configurations (respectively, universal configurations) that use at most n tape cells, and $\to_n$ is the restriction of the next configuration relation to Conf_n.

Let un is an atomic proposition that holds on the universal configurations. Let $ACCEPT$ be the formula $\mu Z.\ (un \wedge \ \Box\Diamond\ Z)$.

Proposition 5 *Let M be an alternating Turing machine.*

1. *The set of accepting universal configurations of K_M is defined by the formula $ACCEPT$.*
2. *If M uses at most n cells on the inputs of length n, then $(K^n_M,\ q_0a_0, \dots a_{n-1}) \models ACCEPT$ iff $a_0, \dots a_{n-1} \in L(M)$.*

Proof: (1) Let Uconf be the set of universal configurations of M. Let $F:\ 2^{\mathrm{Uconf}} \to 2^{\mathrm{Uconf}}$ be the function that maps subsets of the universal configurations to subsets of the universal configurations and is defined as follows:

$$F(S) = \{\alpha \in \mathrm{Uconf}\ |\forall\beta.\ \alpha \to \beta \Rightarrow \exists\beta' \in \mathrm{Uconf}.\ \beta \to \beta' \wedge \beta' \in S\}$$

Notice that F is monotonic, i.e., $S_1 \subseteq S_2$ implies $F(S_1) \subseteq F(S_2)$. It is clear that the set Z of states which satisfies $ACCEPT$ is the least fixed point of F.

According to the Knaster-Tarski theorem Z is equal to $\cup S_i$ where S_0 is the empty set and $S_{i+1} = F(S_i)$. The conclusion of the Proposition follows from the observation that $\alpha \in S_n$ if and only if α is a universal configuration and there is a trace C such that $(\alpha,\ t) \in C$ for some $t < 2n$. The last observation is easily proved by induction as follows.

$\Rightarrow$ Basis is immediate. Inductive step: assume that $\alpha \in S_{n+1}$. Let $\beta_1, \ldots, \beta_k$ be all the configurations such that $\alpha \to \beta_i$. Observe that all β_i are existential. By the definition of S_{n+1} there are $\alpha_i \in S_n$ such that $\beta_i \to \alpha_i$ for $i = 1, \ldots, k$. By induction hypothesis there are accepting traces C_i ($i = 1, \ldots, k$) and $t_i < 2n$ such that $(\alpha_i,\ t_i) \in C_i$ for $i = 1, \ldots, k$. Let C be $\cup_{i=1}^{k} C_i \cup \{(\beta_i, 2n)\ :\ i = 1, \ldots, k\} \cup \{(\alpha, 2n+1)\}$. It is easy to verify that C is an accepting trace and $(\alpha, 2n+1) \in C$. This completes the inductive step.

$\Leftarrow$ Basis is immediate. Inductive step: assume that α is an universal configuration, and there is an accepting trace C and $t < 2n+2$ such that $(\alpha, t) \in C$. We are going to prove that $\alpha \in S_{n+1}$. Let $\beta_1, \ldots, \beta_k$ be all the configurations such that $\alpha \to \beta_i$. Since α is an universal configuration, it follows that β_i (for $i = 1, \ldots, k$) are existential configurations. Since C is an accepting trace, there are $t_1, \ldots t_k < t$ such that $(\beta_i, t_i) \in C$. Since C is accepting and β_i are existential there are universal configurations α_i and $t'_i < t_i$ such that $(\alpha_i,\ t'_i) \in C$ and $\beta_i \to \alpha_i$ for $i = 1, \ldots, k$. Observe that $t'_i < 2n$, because $t'_i < t_i < t < 2n+2$. Moreover, (α_i, t_i) belongs to the accepting trace C. Therefore, by induction hypothesis $\alpha_i \in S_n$ and by the definition of F, we obtain that $\alpha \in F(S_n) = S_{n+1}$. This completes the inductive step.

(2) Observe that if C is a trace and $(\alpha,\ t) \in C$, then the set $\{(\alpha',\ t') \in C \mid t' \leq t$ and α' is reachable from $\alpha\}$ is a trace. Observe also that for such M all the configurations reachable from $q_0 a_0 a_2 \ldots a_{n-1}$ use at most n cells. From these observations and the arguments similar to the arguments used in the proof of (1) we obtain Proposition 5(2).

$\square$

The following proposition shows that K_M^n and $q_0 a_0, \ldots a_{n-1}$ can be represented succinctly by propositional formulas.

Proposition 6 *Let $M = (E,\ U,\ q_0\ \Delta, \Gamma, b,\ \nu)$ be an alternating Turing machine. For every n and a string $a_0 \ldots a_{n-1}$ over the input alphabet of M there are propositional formulas $State$, $Next$ and $Univ$ of size $O(nlog^2 n)$ that describe K_M^n and a propositional formula $Init$ of size $O(nlogn)$ that describes the configuration $q_0 a_0 \ldots a_{n-1}$. Moreover, these formulas are constructed in polynomial time.*

Proof: In order to describe K_M^n we will use the following propositional variables: $tape_{i,\gamma}$ for $i = 0, \ldots, n$ and $\gamma \in \Gamma$, and $contr_{i,q}$ for $i = 0, \ldots, n-1$ and $q \in E \cup U$. Informally $tape_{i,\gamma}$ means that the i-th place of a configuration contains the tape symbol γ and $contr_{i,q}$ means i-th place of a configuration contains the control in the state q. Below we use $contr$ for a sequence of all control variables and $E_1^m(x_1, \ldots x_m)$ for a formula that uses $O(mlogm)$ occurrences of symbols and has the value TRUE iff precisely one of $x_1, \ldots x_m$ has the value TRUE (it was proved in [9] that for every m there exists such a formula; the formula can be defined by recursion as follows: $E_1^{2m}(x_1, \ldots x_{2m}) \triangleq (E_1^m(x_1, \ldots x_m) \wedge \bigwedge_{i=m+1}^{2m} \neg x_i) \vee (E_1^m(x_{m+1}, \ldots x_{2m}) \wedge \bigwedge_{i=1}^{m} \neg x_i)$.

The set of configurations that use at most n-tape cells is represented by the following propositional formula that states that every position i is occupied by exactly one symbol and there is exactly one control variable that has the value TRUE.

$$\begin{array}{ll}
\bigwedge_{i\in\{0,\ldots n-1\}} & (\bigvee_\gamma tape_{i,\gamma} \vee \bigvee_q contr_{i,q}) \wedge \\
& \bigwedge_{\gamma\neq\gamma'}(tape_{i,\gamma} \Rightarrow \neg tape_{i,\gamma'}) \wedge \\
& \bigwedge_{q,\gamma}((contr_{i,q} \Rightarrow \neg tape_{i,\gamma}) \wedge (tape_{i,\gamma} \Rightarrow \neg contr_{i,q})) \\
\bigwedge & \bigwedge_{\gamma\neq\gamma'}(tape_{n,\gamma} \Rightarrow \neg tape_{n,\gamma'}) \wedge (\bigvee_\gamma tape_{n,\gamma}) \\
\bigwedge & E_1^{(n-1)\times|Q|}(contr)
\end{array}$$

This formulas has $O(nlogn)$ occurrences of symbols and these symbols are from the alphabet of size $O(n)$. Hence, the formula can be encoded by a binary string of length $O(nlog^2n)$.

Similarly, it is easy to write formulas $Next$, $Univ$ and $Init$ of size $O(nlogn)$ such that (1) $Next$ specifies the next configuration relation in the Kripke structure K_M^n. (2) $Univ$ says that a configuration is a universal configuration, and (3) $Init$ says that a configuration is the initial configuration $q_0a_0 \ldots a_{n-1}$. $\square$

Finally, Theorems 1 and 3 follow from Theorem 2, and Propositions 5(2) and 6 by the standard arguments.

Remark 7

Succinct Representation of Graphs: Galperin and Wigderson [8] observed that when adjacency matrices of graphs are described by Boolean circuits, there is an exponential blow-up in the complexity of many natural decision problems on graphs. This seminal paper initiated a research on succinct representation of graphs and other structures.

Succinctness of formalisms for Concurrent Systems: There are many specification formalisms for concurrent systems. A concurrent system is assembled from simpler sequential agents by a parallel composition. There is a variety of definitions of parallel composition which reflect alternatives: message passing vs shared variables, synchronous vs asynchronous.

Sequential components are usually described by Kripke structures (equivalently, by labeled transition systems or by automata). A semantics defines what is the Kripke structure assigned to the parallel composition of structures $p_1, \ldots, p_n$ (see, e.g [4]). If each component structure p_i is finite, then their parallel composition $q = p_1||p_2|| \ldots ||p_n$ is finite. However, the number of states in q is of the order $|p_1| \times |p_2| \ldots \times |p_n|$ and is exponential in $|p_1| + |p_2| + \ldots + |p_n|$ - the size of its description. The model checking

problem for such a formalism can be defined as follows:
Input: finite state structures $p_1, \dots, p_n$, and a μ-calculus formula ϕ.
Question: Determine whether the structure $p_1||p_2 \dots ||p_n$ satisfies the property defined by ϕ.

Let M be a linear space alternating Turing machine. For every n the structure K_M^n can be succinctly described in every reasonable formalism for the description of concurrent systems. For example, it is easy to give a description of size $O(nlogn)$ for K_M^n in terms of asynchronous concurrent systems of finite automata (see e.g., [13]); one can provide a synchronous or Boolean circuit of size $O(nlogn)$ [8] or a systolic array of size $O(n)$ that describes K_M^n. Therefore, it follows that the problem of checking whether an asynchronous concurrent system of finite automata (respectively a synchronous circuit, or a systolic array) satisfies the formula $ACCEPT$ cannot be solved in deterministic time $O(c^{n/logn})$ (respectively, $O(c^{n/logn})$ or $O(c^n)$).

OBDD: Ordered binary decision diagrams (OBDD) are widely used in computer-aided verification for digital design, verification and testing [1, 12]. In [6] the complexity of many decision problems on graphs represented by OBDD was investigated. From the proof of Theorem 10 in [6] one can extract a polynomial in n OBDD description of K_M^n. Therefore, the following problem is EXPTIME complete:
Input: A description of a finite state Kripke structure K and its state s by OBDDs, and a μ-calculus formula ϕ.
Question: Determine whether the state s of the Kripke structure K satisfies the property defined by ϕ.

After this work had been completed the author learned that in [11] it was proved that the program complexity of μ-calculus model checking for concurrent programs is EXPTIME-complete. In the proof of the above theorem Kripke structures are described by finite state concurrent programs. The input to model checker is a concurrent program and a μ-formula. It is proven in [11] that for every linear space alternating Turing machine M and an input $a_0 \dots a_n$ one can construct a program Pr of length n and an alternation free μ-formula ϕ of length n such that M accepts $a_0 \dots a_n$ iff the Kripke structure described by Pr satisfies ϕ.

Acknowledgments

I would like to thank a referee for his helpful suggestions and Moshe Vardi for pointing out to the papers [6, 11].

References

[1] J. Burch, E. Clarke, k¿ McMillan, D. Dill and L. Hwang. Symbolic model checking: 10^{20} states and beyond. Information and Computation98(2):142-170, 1002.

[2] A. Chandra, D. Kozen and L. Stockmeyer. Alternation. Journal of the Association for Computing Machinery, 28:114-131, 1981.

[3] E. Clarke and E. Emerson. Design and Synthesis of Synchronization Skeleton Using Branching Time Temporal Logic. In *Proceedings of Workshop on Logic of Programs*, LNCS vol. 131, 1981.

[4] E. Emerson. Temporal and Modal Logic, in *Handbook of theoretical computer science*, 1990,997-1072.

[5] E. Emerson and C. Lei. Efficient model checking in fragment of the propositional mu-calculus. In *Proceedings of the First Annual Symposium on Logic in Computer Science*, 1986.

[6] J. Feigenbaum, S¿ Kannan, M. Vardi, M. Viswanathan. Complexity of Problems on Graphs Represented as OBDDs. Proc. of 15-th STACS. LNCS vol. 1373, Springer Verlag, 1998, pp 216-226.

[7] M. Fisher and R. Lander. Propositional Dynamic Logic of Regular Programs. Journal of Computer and System Sciences, 18:194-211, 1979.

[8] H. Galperin and A¿ Wigderson. Succinct Representation of Graphs. Information and Control 56 :183-198, 1983.

[9] L. Khasin. Complexity bounds for the realizations of monotonic symmetrical functions by means of formulas in the basis $\{\wedge, \vee, \neg\}$. Soviet Physics Doklady, 14:1149-1151, 1970.

[10] D. Kozen. Results on the propositional mu-calculus. Theoretical Computer Science, 27:33-354, 1983.

[11] O. Kupferman, M. Vardi and P. Wolper. Automata theoretical approach to branching time model checking. Manuscript, June 1998.

[12] K. McMillan. Symbolic Model Checking, Ph.D thesis, CMU, 1992.

[13] A. Rabinovich. Complexity of equivalence problems for concurrent systems of finite agents. Information and Computation 139 (1997), 111-129.

[14] C. Stirling. Modal and temporal logics. In *Handbook of Logic in Computer Science*, Oxford University press, 1992, 476-567.