МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2003
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2003
Proceedings of the Thirty Second Spring Conference of
the Union of Bulgarian Mathematicians
Sunny Beach, April 5–8, 2003

OPEN SYSTEMS VERSUS CLOSED SYSTEMS

Irena L. Atanasova

While linear-time and branching-time temporal logics (LTL and BTL) are natural specification languages for closed systems, alternating-time logic ATL is natural specification language for open systems. LTL assumes implicit universal quantification over all paths that are generated by the execution of a system. BTL allows explicit existential and universal quantification over all paths. ATL offers selective quantification over those paths that are possible outcomes of games. In Table 1 are shown the model-checking problem complexity results for BTL and ATL logics.

1. Introduction. In 1977, Pnueli proposed to use linear-time temporal logic (LTL) to specify requirements for reactive systems. A formula of LTL is interpreted over a computation, which is an infinite sequence of states. Branching-time temporal logic such as CTL and CTL* do provide explicit quantification over the set of computations [6]. The problem of model checking is to verify whether a finite-state abstraction of a reactive system satisfies a temporal-logic specification [4]. Efficient model checkers exist for LTL, CTL and CTL* [11, 13]. The logics LTL and CTL are interpreted over Kripke structure. An open system is a system that interacts with its environment and whose behavior depends on the state of the system as well as the behavior of the environment. Modeling languages for open systems distinguish between internal nondeterminism (system), and external nondeterminism (environment). Besides universal and existential questions, there is the third: can the system resolve its internal choices. Such an alternating satisfaction can be viewed as a winning condition in a 2-player game between the system and the environment.

In this paper, we consider the linear-time temporal logic, branching-time temporal logics and alternating-time temporal logics. We do not consider the basic definition for LTL, CTL, CTL*, ATL and ATL*, and we establish and summarize complexity bounds on model checking for these logics.

2. Model-checking complexity. Lichtenstein and Pnueli argued that when analyzing the complexity of model checking, a distinction should be made between complexity in the size of the input structure and complexity in the size of the input formula. We measure the complexity of the model-checking problem in two different ways: the program complexity and the full complexity of model checking. Since the structure is typically much larger than the formula, and its size is the common computational bottleneck, the program-complexity measure is of particular practical interest. There is the third kind of complexity – time complexity.

Closed systems

Theorem 2.1. *The program complexity of model checking for CTL and CTL* is NLOGSPACE-complete.*

Proof. Let the formula is fixed. We get a hesitant alternating automaton HAA of a fixed depth. The nonemptiness problem for such a HAA is in NLOGSPACE [8]. Thus the program complexity of CTL and CTL* model checking is the same. Hardness in NLOGSPACE is immediate by a reduction from the graph accessibility problem, proved to be NLOGSPACE-complete in [7].

Open systems

Consider a game structure S=(k, Q, P, π, d, δ) and a set A$\subseteq \Sigma$ of players. We build the following 2-player turn-based synchronous game structure $\mathrm{S_A} = (2, \mathrm{Q_A}, \mathrm{P_A}, \pi_\mathrm{A}, \sigma_\mathrm{A}, \mathrm{R_A})$. If the game structure S has m transitions, then the turn-based synchronous structure $\mathrm{S_A}$ has O(m) states and transitions.

Lemma 2.1. *Let* S *be a game structure with state space* Q, A *be a set of players of* S *and* p *be a proposition of* S. *Then*

$$[\ll \mathrm{A} \gg .\mathrm{p}]_\mathrm{S} = [\ll 1 \gg .\mathrm{p}[_{\mathrm{SA}} \cap \mathrm{Q} \quad \text{and} \quad [\ll \mathrm{A} \gg .\mathrm{p}]_\mathrm{S} = [\ll 1 \gg .(\mathrm{p} \vee \mathit{newstate})]_{\mathrm{SA}} \cap \mathrm{Q},$$

where newstate is a special proposition that identifies the new states: $\mathrm{P_A} = \mathrm{P} \cup \{\mathit{newstate}\}$.

Consider the game structure S and fairness condition G for S. We define the following extended game structure: $\mathrm{S^F} = (\mathrm{k}, \mathrm{Q^F}, \mathrm{P^F}, \pi^\mathrm{F}, \mathrm{d^F}, \delta^\mathrm{F})$, where $\mathrm{Q^F} = \{(\mathit{false}, \mathrm{q}) | \mathrm{q} \in \mathrm{Q}\} \cup \{(\mathrm{q}', \mathrm{q}) | \mathrm{q}$ is a successor of q' in S$\}$; for $(a, \gamma) \in \mathrm{G}$ there are new propositions $(a, \gamma, \mathit{allowed})$ and $(a, \gamma, \mathit{taken})$, that are $\mathrm{P^F} = \mathrm{P} \cup (\mathrm{G}\{\mathit{allowed}, \mathit{taken}\})$, where the new propositions allow us to identify the fair computations; for the $(\mathit{false}, \mathrm{q}) \in \mathrm{Q^F}$, we have $\pi^\mathrm{F}((\mathrm{false}, \mathrm{q})) = \pi(\mathrm{q})$, for $(\mathrm{q}', \mathrm{q}) \in \mathrm{Q^F}$ we have

$$\pi^\mathrm{F}((\mathrm{q}', \mathrm{q})) = \pi(\mathrm{q}) \cup \{(a, \gamma, \mathit{allowed}) | \gamma(\mathrm{q}') \neq \emptyset\} \cup \{(a, \gamma, \mathit{taken}) |$$

there is a $(\mathrm{j}_1, \ldots, \mathrm{j}_\mathrm{k}) \in \mathrm{D}(\mathrm{q}')$ such that $\mathrm{j}_a \in \gamma(\mathrm{q}')$ and $\delta(\mathrm{q}', \mathrm{j}_1, \ldots, \mathrm{j}_\mathrm{k}) = \mathrm{q}\}$; for $a \in \Sigma$ and $(\cdot, \mathrm{q}) \in \mathrm{Q^F}$, we have $\mathrm{d}_a^\mathrm{F}((\cdot, \mathrm{q})) = \mathrm{d}_a(\mathrm{q})$; for $(\cdot, \mathrm{q}) \in \mathrm{Q^F}$ and $(\mathrm{j}_1, \ldots, \mathrm{j}_\mathrm{k}) \in \mathrm{D}(\mathrm{q})$, we have $\delta^\mathrm{F}((\cdot, \mathrm{q}), \mathrm{j}_1, \ldots, \mathrm{j}_\mathrm{k}) = \delta(\mathrm{q}, \mathrm{j}_1, \ldots, \mathrm{j}_\mathrm{k})$.

Lemma 2.2. *A state* q *of the game structure* S *fairly satisfies a Fair ATL formula of the form* $\ll \mathrm{A} \gg \Psi$, *where* A *is a set of players of* S, *and* $\Psi = \mathrm{p}_1 \mathrm{U} \mathrm{p}_2$ *or* $\Psi = \nabla \mathrm{p}$, *with respect to the weak-fairness condition* $\mathrm{G_W}$ *iff the state* $(\mathit{false}, \mathrm{q})$ *of* $\mathrm{S^F}$ *satisfies the following ATL* formula:* $\ll \mathrm{A} \gg (\wedge_{a \in \mathrm{A}, (a,\gamma) \in \mathrm{G_W}} \nabla \diamondsuit (\neg(a, \gamma, \mathit{allowed}) \vee (a, \gamma, \mathit{taken})) \wedge (\wedge_{a \in \Sigma \setminus \mathrm{A}, (\mathrm{a},\gamma) \in \mathrm{G_W}} \nabla \diamondsuit (\neg(a, \gamma, \mathit{allowed}) \vee (a, \gamma, \mathit{taken})) \rightarrow \Psi))$.
A state q *fairly satisfies* $\ll \mathrm{A} \gg \Psi$ *with respect to the strong-fairness condition* $\mathrm{G_S}$ *iff* (false, q) *satisfies the following ATL* formula:* $\ll \mathrm{A} \gg (\wedge_{a \in \mathrm{A}, (a,\gamma) \in \mathrm{G_S}} \nabla \diamondsuit (a, \gamma, \mathit{allowed}) \rightarrow \nabla \diamondsuit (a, \gamma, \mathit{taken})) \wedge (\wedge_{a \in \Sigma \setminus \mathrm{A}, (a,\gamma) \in \mathrm{G_S}} (\nabla \diamondsuit (a, \gamma, \mathit{allowed}) \rightarrow \nabla \diamondsuit (a, \gamma, \mathit{taken})) \rightarrow \Psi))$.

Theorem 2.2. *The model-checking problem for ATL can be solved in time* O(m.l) *for a game structure with m transitions and an ATL formula of length* l.

Proof. Consider a game structure S with m transitions and an ATL formula φ of length l.

Consider the following symbolic algorithm for ATL model checking, which manipulates state sets of S:
for (each φ' in Subformula (φ))

switch (φ') {case φ'=p:[φ']=state_sat_p (p);
case φ'=$\neg\theta$: [φ']=[true]\[θ];
case φ'=$\theta_1 \vee \theta_2$: [φ']= [θ_1]$\cup$[θ_2];
case $\varphi' = \ll$A$\gg \otimes \theta$: [φ']=Player_state(A, [θ]);
case $\varphi' = \ll$A$\gg\nabla\theta$: ρ=[true];τ=[θ];
while ($\rho\Box\tau$)
{ρ=$\rho \cap \tau$;τ=Player_state(A, ρ)$\cap$[θ];}[φ']= ρ;
case $\varphi' = \ll$A$\gg\theta_1 \cup \theta_2$: ρ=[false];τ=[θ_2];
while ($\tau.\rho$)
{ρ=$\rho \cup \tau$;τ=Player_state(A, ρ)$\cap$[θ_1];} [φ']= ρ;}
return [φ];

We claim that this algorithm can be implemented in time O(m.l). The size of Subformula (φ) is bounded by l. To compute [$\ll$A$\gg.\varphi$] from [φ], we apply the second part of Lemma 2.1. The resulting invariance game on S_A can be solved in time linear in the size of S^A, that is, in time O(m) [3]. To compute [$\ll$A$\gg\theta_1 \cup \theta_2$] from [$\varphi_1$] and [$\varphi_2$], we first restrict the game structure S to the states in [φ_1] $\cup$ [φ_1], and then apply the first part of Lemma 2.1. The resulting reachability game can again be solved in time O(m).

Theorem 2.3 [1]**.** *The model-checking problem for ATL is PTIME-complete. The problem is PTIME-hard even for a fixed formula.*

Theorem 2.4. *The model-checking problem for Fair ATL is PSPACE-complete, and can be solved in time $m^{O(w)}.l$ for a game structure with m transitions, w fairness constraints, and a Fair ATL formula of size l. The problem is PSPACE-hard even for a fixed formula. For a bounded number of fairness constraints, the problem is PTIME-complete.*

Proof. The model-checking algorithm labels each state of the S^F with all subformulas of given Fair ATL formula φ, starting with the innermost subformulas. The interesting case is when the subformulas are of the form $\ll$A$\gg\Psi$. This requires solving a game on S^F with the winning condition of the form given by the second part of Lemma 2.2. In [2], it is shown that turn-based games whose condition is a Boolean combination of formulas of the form $\Box\Box$p can be solved in PSPACE, or in time m^n, where n is the size of the formula and m is the size of the game structure. The size of the winning condition is O(w), where w is the number of fairness constraints. Therefore, each temporal operator can be processed in time $m^{O(w)}$, leading to the overall complexity bound.

For the lower bounds, the construction of [2] can be modified to reduce the satisfaction of a given quantified Boolean formula Φ to Fair ATL model checking of a fixed formula of the form $\ll a\gg\nabla p$ over 2-player turn-based synchronous game structure of size $O(|\Phi|)$.

Theorem 2.5. *The model-checking problem for ATL* is 2EXPTIME-complete. The model-checking problem for ATL* formulas of bounded size is PTIME-complete.*

Proof. As in the algorithm for CTL* model checking, we label each state q of S by all state subformulas of φ that are satisfied in q, starting from the innermost state subformulas of φ. Let $\varphi' = \ll$A$\gg\Psi$. We construct a Rabin tree automaton Aφ that accepts precisely the trees that satisfy the CTL* formula $\forall\ \Psi$. For each state q of S, we construct a Büchi tree automaton $A_{S,q,A} = (2^\pi, Q, \eta, q, F)$ that accepts the (q, A)-execution trees. The product of the two automata is a Rabin tree automaton that accepts

the (q, A)-execution trees that satisfy $\forall\Psi$. The size of the $A_{S,q,A}$ is bounded by the size of game structure S. The nonemptiness problem for a Rabin tree automaton of size n with r Rabin pairs can be solved in time $O(n.r)^{3r}$ [12]. Labeling a single state with φ' requires at most time $\left(|S|.2^{2^{O(|\Psi|)}}\right)^{2^{O(|\Psi|)}} = |S|^{2^{O(|\Psi|)}}$. Since there are |Q| states and at most $|\varphi|$ subformulas, membership in 2EXPTIME follows.

The lower bound for the program complexity follows from Theorem 2.2 and theorem 2.3, and the upper bound from fixing $|\Psi|$ in the analysis of the joint complexity above.

Systems/ Logic	Complexity		
	full	program	time
Open systems			
ATL	PTIME [1]	PTIME [1]	O(m.l)
Fair ATL	PSPACE	PSPACE	$m^{O(w)}$
ATL*	2EXPTIME	PTIME	$m^{2^{O(l)}}$ [1]
Closed systems			
CTL	PTIME [5]	NLOGSPACE [10]	$O(m.\log^2(m.n))$ [8]
Fair CTL	PTIME [5]	PTIME [9]	$O(m.\log^2(n.l))$ [8]
CTL*	PSPACE [5]	NLOGSPACE [10]	$O(m(m + \log n)^2)$ [8]

Table 1

3. Conclusions. While closed systems are naturally modeled as Kripke structures, a general model for open systems is the concurrent game structure. Closed systems correspond to the special case of a single player. Game structure degenerates to Kripke structure, ATL degenerates to CTL and ATL* to CTL* in this case. The model – checking complexity results are summarized in Table 1. All complexities in the table denote tight bounds. m is the size of the structure and l is the length of the formula.

REFERENCES

[1] R. Alur, Th. Henzinger, O. Kupferman. A Hernating-time Temporal Logic. Proceedings of the 38th IEEE Symposium on Foundations of Computer Science (FOCS), 1997, 100–109.

[2] R. Alur, S. La Torre, P. Madhusudan. Playing Games with Boxes and Diamonds. Technical report, University of Pennsylvania, 2002.

[3] C. Beeri. On the membership problem for functional and multi-valued dependencies in relational databases. *ACM Transactions on Database Systems*, **5** (1980), 241–259.

[4] E. M. Clarke, E. A. Emerson. Design and synthesis of synchronization skeletons using branching-temporal logic. In: Proceedings of the International Workshop on Logic of Programs. Springer-Verlag, **131**, 1981, 52–71.

[5] E. M. Clarke, E. A. Emerson, A. P. Sistla. Automatic Verification of finite-state concurrent systems using temporal-logic specifications. *ACM Transactions on Programming Languages and Systems*, **8** (1986), No 2, 244–263.

[6] E. A. Emerson, J. Y. Halpern. Sometimes and not never revisited: On branching versus linear time. *Journal of the ACM*, **33** (1986), No 1, 151–178.

[7] N. D. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, **11** (1975), 68–75.
[8] O. Kupferman. Model Checking for Branching-Time Temporal Logic. PhD Thesis, 1995.
[9] O. Kupferman, M. Vardi. Verification of fair transition systems. *Chicago Journal of Theoretical Computer Science*, (1998).
[10] O. Kupferman, M. Vardi, P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, **47** (2000), No 2, 312–360.
[11] K. L. McMillan. Symbolic Model Checking. Kluwer Academic Publishers, 1993.
[12] A. Pnueli, R. Rosner. On the synthesis of a reactive module. In: Proceedings of the 16th International Symposium on Principles of Programming Languages. ACM Press, 1989, 179–190.
[13] W. Visser. Efficient CTL* Model Checking Using Games and Automata. PhD Thesis, 1998.

Irena Atanasova
Southwestern University "N. Rilski"
Department of Computer Science
66, Ivan Mihajlov Str.
2700 Blagoevgrad, Bulgaria
e-mail: irenatm@aix.swu.bg

ОТВОРЕНИТЕ СИСТЕМИ СРЕЩУ ЗАТВОРЕНИТЕ СИСТЕМИ

Ирена Атанасова

Докато линейната във времето и разклонената във времето темпорални логики LTL и BTL са естествени езици за спецификация за затворени системи, то алтернативната във времето логика ATL е естествен език за спецификация за отворени системи. LTL допуска само квантификация на общност над всички пътища, които са генерирани чрез изпълнението на система. BTL позволява квантификация за общност и съществуване над всички пътища. ATL предлага квантификация по избир над тези пътища, които са възможни резултати от игрите. В таблица 1 са показани резултатите за сложността на проблема на проверката на модела за BTL и ATL логиките.