Bull. Nov. Comp. Center, Comp. Science, 1 (1993)

An effective model checking for Mu-calculus: from finite systems towards systems with real time

S.A. Berezin and N.V. Shilov and P.V. Shneider

Mu-calculus [1] is a polymodal logic with fixed points. A Decision Procedure (DP) checks the validity of a formula. A Model-Checking Procedure (MCP) constructs the validity set of a formula in a model. Since Mu-calculus is finitely approximatizible [1] and it is applicable for verification of Finite-State Machines an effective MCPs are of high importance [2]. The paper deals with the direct exponential MCP and a polynomial approximation of MCP, which is correct for a representative fragment of Mu-calculus on finite models, and with an extention of a MCP for finite models to infinite models with a real time.

1. The syntax and semantics of Mu-calculus. The model-cheking problem for Mu-calculas

Let p, q, r, ...be the alphabet of predicate symbols (or predcates), x, y, z, ...be the alphabet of (logical) variables, a, b, c, ...be the alphabet of action symbols (or action). All those alphabets are countable and disjoint.

Definition 1. The syntax of Mu-calculus consists of formulae.

1. For a predicate p the expression "p" is a formula without free and bound variables.
2. For a variable x the expression "x" is a formula with the unique free variable x itself and without bound variables.
3. For a formula W the negation "$\neg W$" is a formula with the same free and bound variables as W.
4. For a finite set of formulae $\mathcal{F}$ if there is no variable which is free and bound in the formulae from $\mathcal{F}$, the conjunction "$\bigwedge \mathcal{F}$" and the disjunction "$\bigvee \mathcal{F}$" are formulae with sets of free and bound variables equal to the union of free and bound variables respectively of formulae from $\mathcal{F}$.

5. For a formula W and a variable x, if x is free variable of W and all occurrences of x in W are under even number of negations, the expressions "$min \quad x.W$" for the least fixed point and "$max \quad x.W$" for the greatest fixed point are formulae with the leading variable x and the sets of free variables equal to the same ones of W without x and the sets of bound variables equal to the same ones of W with addition of x.

6. For a formula U and a action b the expressions "$[b]\,U$" and "$< b > U$" are formulae with the same free and bound variables as U.

For a formula W and a list of variables Var without doubles we will write "$W(Var)$" iff any free variable of W occurs in Var and all bound variables of W are absent in Var.

A model M is a pair of the form (D_M, I_M), where D_M is a non-empty set of states $s, t, \ldots$ and I_M is a mapping such that:

1. for any predicate $I_M(p) \subseteq D_M$;

2. for any action $Im(a) \subseteq D_M \times D_M$.

Definition 2. If M is a model and $V(Var)$ is a formula, then $\lambda E.[V, E]_M$ is a general function from $(2^{D_M})^{|Var|}$ to 2^{D_M} which means that for any evaluation E of free variables by subsets of D_M the $[V, E]_M \subseteq D_M$; the set $[V, E]_M$ is defined below.

1. For a predicate p, if $V = p$, then $[V, E]_M = I_M(p)$.

2. For a variable x, if $V = x$, then $[V, E]_M = E(x)$.

3. For any formula W, if $V = \neg W$ then $[V, E]_M = D_M \setminus [W, E]_M$.

4. For a set of formulae $\mathcal{F}$

   (a) if $V = \bigwedge \mathcal{F}$, then $V_M = \bigcap_{W \in \mathcal{F}} [W, E]_M$;

   (b) if $V = \bigvee \mathcal{F}$, then $V_M = \bigcup_{W \in \mathcal{F}} [W, E]_M$.

5. For a formula W and a variable x, if $V = fix \quad x.W$, where $fix \in \{min, max\}$, then the list of variables (Var, x) is a list of free variables of W; let $Fix(W, M, E)$ be $\{set \subseteq D_M | [W, E(set/x)]_M = set\}$ with the partial order $\subseteq$;

   (a) if $V = min \quad x.W$, then

   $$[V, E]_M = \text{ the least set from } Fix(W, M, E);$$

(b) if $V = max \quad x.W$, then

$$[V,E]_M = \text{ the greatest set from } Fix(W,M,E).$$

6. For a formula W and a action b

(a) if $V = [b]W$, then

$$[V,E]_M = \{s|\text{for all} \quad t \quad \text{if } (s,t) \in I_M(b) \text{ then } t \in [W,E]_M\};$$

(b) if $V =< b > W$, then

$$[V,E]_M = \{s| \text{ there exists} \quad t \quad \text{such that}$$

$$(s,t) \in I_M(b) \quad \text{and} \quad t \in [W,E]_M\}.$$

The Model-Checking Problem for Mu-Calculus is an algorithmic problem of how to construct the semantic set $[V,E]_M$ for a formula V on a model M and an evaluation E for free variables. A Model-Checking Procedure is an algorithm for the Model-Checking Problem.

Statement 1. *For a partial-ordered set S with the least $\bot$ and the greatest $\top$ elements, and a non-descending general function $f : S \mapsto S$ the following hold:*

1.1. if the least fixed point of f exists, then for all $n \geq 0$ the value $f^n(\bot)$ is not greater than the least fixed point of f;

1.2. if there exists a natural number $n \geq 0$ such that $f^n(\bot) = f^{n+1}(\bot)$, then the least fixed point of f exists and is equal to $f^n(\bot)$;

1.3. if the greatest fixed point of f exists, then for all $n \geq 0$ the greatest fixed point is not less than $f^n(\top)$;

1.4. if there exist a natural number $n \geq 0$ such that $f^n(\top) = f^{n+1}(\top)$, then the greatest fixed point exists and is equal to $f^n(\top)$.

Lemma 1. *For a finite model the semantics of a formula is correctly defined: the semantics of a formula is a non-descending function on variables with positive instances only and non-ascending on variables with negative instances only.*

Remark. An instance of a subformula in formula is said to be positive/negative iff it is under even/odd number of negations.

SKETCH OF PROOF. Let us fix a model M and design a sketch of a proof by induction on the structure of a formula. The base of the induction — the case when a formula is a predicate or a variable — is obvious. The induction step in the case when a formula is a boolean combination of subformulae is trivial too, but it is important that the negation transforms positive instances into negative and vice versa.The induction step in the case when a formula is a modal formula is also trivial, but let us consider the case when a formula is a modal formula of the form: $< b > W$, where b is an action and W is a subformula.

In this case for an evaluation E of free variables of V we have: $[< b > W, E]_M = \{s|$, there exists t such that $(s,t) \in I_M(b)$ and $t \in [W,E]_M\}$ so, for any evaluations E' and E'', if for any variable x with positive/negative instances only $E'(x) \subseteq / \supseteq E''(x)$, then $[W,E']_M \subseteq [W,E'']_M$ and hence $[< b > W, E']_M \subseteq [< b > W, E'']_M$. The induction step in the case when a formula is a fixed point formula is of the most importance, but let us consider the case when a formula is a fixed point formula of the form "$min \quad x.W$", where x is a variable and W is the subformula. In this case for a list sets of subsets of D_M we have (in accordance with Statement 1 and because of D_M finiteness) the least set of $\{set \subseteq D_M| \quad [W, E(set/x)]_M = set\}$ is equal to $\bigcup_i set_i$, where $set_0 = \emptyset$ and for each $i \geq 0 \quad set_{i+1} = [W, E(set_i/x)]_M$. So, for any evaluations E' and E'' if for any variable y with positive/negative instances only $E'(y) \subseteq / \supseteq E''(y)$ then for all $i \geq 0 \quad set'_i \subseteq set''_i$ and $[min \quad x.W, E']_M = \bigcup_i set'_i \subseteq \bigcup_i set''_i = [min \quad x.W, E'']_M$. □

Corollary 1. *The Model-Checking Problem is decidable with the upper bound* $(const \times m^3 \times f)^{n+1}$, *where* m *is the capacity of the domain of a model,* f *is the length of a formula and* n *is the maximal depth of nested fixed points.*

SKETCH OF PROOF. Let us describe a recursive procedure which will be referred to as the Direct Model-Checking Procedure (DMCP). The DMCP is applicable to triples of the form (M,V,E), where M is a finite model, V is a formula and E is an evaluation for free variables and for each such triple $DMCP(M,V,E)$ should be equal to $[V,E]_M$. The definition of the $DMCP$ almost coincides with the definition of the semantics of Mu-calculus, but the fifth step:

(5'.a) if $V = min \quad x.W$ then

```
begin
  i := 0; set_i := ∅ ;
  repeat
```

$i := i + 1$;
$E_i := E(set_{i-1}/x)$
$set_i := DMCP(M, W, E_i)$;
until $set_i \neq set_{i-1}$
end;
$DMCP(M, V, E) := set_i$;

(5'.b) if $V = max \quad x.W$ then

begin $i := 0$; $set_i := D_M$;
repeat
$i := i + 1$;
$E_i := E(set_{i-1}/x)$
$set_i := DMCP(M, W, E_i)$;
until $set_i \neq set_{i-1}$
end;
$DMCP(M, V, E) := set_i$.

So, the time complexity bound $TCB(M, V, E)$ for $DMCP(M, V, E)$ can be approximated in accordance with the steps of the DMCP definition as follows:

1. $const \times |D_M|$;
2. $const \times |D_M|$;
3. $const \times |D_M| + const \times TCB(M, W, E)$;
4. $const \times |D_M| + const \times (\Sigma_{W \in \mathcal{F}} TCB(M, W, E(D_M/x)))$;
5. (because of the finiteness of M, bodies of loops are calculated at most $|D_M|$ times)

$$const \times |D_M| \times (TCB(M, W, Sets \cup \{x\}));$$

6. $const \times |Dm|^2 \times |Dm| + const \times (TCB(M, W, E))$;

Obviously, a function $(const \times m^3 \times f)^{n+1}$ majorettes $TCB(M, V, E)$, where $m = |D_M|$, $f = |V|$ and n is the maximal depth of nested fixed points. □

2. The decidability of the model-cheking problem for Mu-calculus

A normal formula is a fixed point formula in which the negation is applied to predicates and variables only.

Lemma 2. *There exists a procedure with a linear upper time bound which transforms any formula in an equivalent normal formula.*

SKETCH OF PROOF. The process of normalization is based on the Morgan's laws and obvious equivalences:

1. for a action b and a formula V formulae $\neg([b]V)$ and $\neg(< b > V)$ are equivalent to $< b > (\neg V)$ and $[b](\neg V)$ respectively;
2. for a variable x and a formula V formulae $\neg(min \quad x.V)$ and $\neg(max \quad x.V)$ are equivalent to $max \quad x.(\neg V(\neg x/x))$ and $min \quad x.(\neg V(\neg x/x))$ respectively.

At the same time each formula V is equivalent to a closure $min \quad x.V$, where x is a variable which does not occur in V.

An equation is the expression of the form $x = V$, where x is a variable and V is a formula without fixed points. □

Let us describe a recursive algorithm of a decomposition of normal formulae into pairs of systems of equations. For a formula V this algorithm constructs two systems of equations $MIN(V)$ and $MAX(V)$ as follows:

$MIN := MAX := \emptyset$; $EQ(V, MIN, MAX)$;
$MIN(V) := MIN$; $MAX(V) := MAX$.
where
procedure EQ (V : a normal formula;
var MIN, MAX : systems of equations);
begin
(* let's present V as $fix \quad x.W(V[1 \ldots k])$, where $fix \in \{min, max\}$, x is a leading variable of V, W is a fixed points free context and $V[1 \ldots k]$ is a vector of normal subformulae of V; let $x[1 \ldots k]$ be the vector of leading variables of $V[1 \ldots k]$; let eq be the equation $x = W(x[1 \ldots k])$;*)
if $fix = min$ then
$MIN := MIN \cup \{eq\}$;
if $fix = max$ then

$MAX := MAX \cup \{eq\}$;
for $i := 1$ to k do $EQ(V_i, MIN, MAX)$;
end.

Statement 2. *The decomposition procedure has a linear time bound.*

Let us describe an iterative algorithm of an approximation for some solution of pairs of systems of equations in a finite model and for an evaluation of variables which do not occur in the left parts of equations (so called free variables). For a pair S of systems of equations

$$\begin{cases} MIN : x_i = F_i(x[1\ldots f], y[1\ldots g], z[1\ldots h]), \\ MAX : y_j = G_j(x[1\ldots f], y[1\ldots g], z[1\ldots h]), \end{cases}$$

where $i \in [1\ldots f]$, $j \in [1\ldots g]$, $x[1\ldots f], y[1\ldots g], z[1\ldots h]$ are disjoint vectors of variables, for a finite model $M = (D_M, I_M)$ and for an evaluation E of free variables $z[1\ldots h]$ by subsets of D_M this algorithm tries to construct subsets of D_M

$$x_i(S, M, E), \quad y_j(S, M, E), \quad z_k(S, M, E)$$

$$i \in [1\ldots f], \quad j \in [1\ldots g], \quad k \in [1\ldots h]$$

as follows:

$s := 0$;
let $E(s)$ *be the evaluation* $E(\emptyset/x[1\ldots f])$
repeat
  $t := 0$;
  let $E(s,t)$ *be the evaluation* $E(s)(D_M/y[1\ldots g])$;
  repeat
    let $E(s,t+1)$ *be the evaluation*
    $E(s,t)([G[1\ldots g], E(s,t)]_M/y[1\ldots g])$;
    $t := t+1$
  until $E(s,t) = E(s,t-1)$;
  let $E(s+1)$ *be the evaluation*
  $E(s,t)([F[1\ldots f], E(s,t)]_M/x[1\ldots f])$;
  $s := s+1$
until $E(s) = E(s-1)$;
the evaluation $E(s)$ *defines values of*
$x_i(S, M, E)$, $y_j(S, M, E)$, $z_k(S, M, E)$
$i \in [1\ldots f]$, $j \in [1\ldots g]$, $k \in [1\ldots h]$

Lemma 3. *The Approximation Procedure has upper time bound const* $\times$ $(m \times f)^4$ *, where* m *is the capacity of the domain of a finite model and* f *is the length of systems of equations.*

Remark. The length of systems of equations is the sum of lengths of all equations from these systems.

SKETCH OF PROOF. Let M be a finite model, S — a pair of systems MIN and MAX of equations and E — an evaluation of free variables of S. The semantics of any fixed-point free formula H from the system S with variables $x[1 \ldots f], y[1 \ldots g]$ and $z[1 \ldots h]$ for any evaluation E' of $x[1 \ldots f]$, any evaluation E'' of $y[1 \ldots g]$ and for the evaluation E of $z[1 \ldots h]$ can be computed in the time *const* $\times$ $m^2 \times h$, where h is the length of H. Hence, each computation of the body of inner loop of the Approximation Procedure can be done in the time *const* $\times$ $m^2 \times f^2$. But the negation in formulae $G_1, \ldots, G_g$ is applied to the variables $z[1 \ldots h]$ only; consequently, $G_1, \ldots, G_g$ are non-descending functions on $x[1 \ldots f]$ and $y[1 \ldots g]$. Hence, because of M finiteness, the body of the inner loop is computed not more than $m \times f$ times. So, each computation of the inner loop can be done in the time *const* $\times$ $m^3 \times f^3$. Obviously, this is upper bound for each computation of the outer loop body of the Approximation Procedure. But formulae $F_1, \ldots, F_f$ are similar to $G_1, \ldots, G_g$. Hence, because of M finiteness, the outer loop body is computed not more than $m \times f$ times too. So, the upper time bound for the Approximation Procedure is *const* $\times$ $m^4 \times f^4$. □

Statement 3. *For any finite model, for any normal formula and for any evaluation of free variables of the formula if the least fixed points of the formula are syntactically independent of outer greatest fixed points of the formula then the Approximation Procedure to be applied to the decomposition of the formula into the pair of systems of equations, and to the model, and to the evaluation, evaluates the semantics of all those fixed points correctly.*

Remark. If V is a formula , $fix_1 \quad x_1.W_1$ is a subformula of V, $fix_2\ x_2.W_2$ is a subformula of W_1, then the last subformula is said to be syntactically independent of the first subformula in V iff x_1 does not occur in W_2.

Theorem 1. *The model-checking problem for Mu-Calculus formulae, for finite models and evaluations of free variables is decidable with an exponential upper time bound; but if a formula is a normal formula in which each least*

fixed point is syntactically independent of any outer greatest fixed point, then the model-checking problem has a polynomial upper time bound.

SKETCH OF PROOF. An exponential upper time bound in the general case is proved in Corollary 1 from Lemma 1. A polynomial upper time bound in the special case follows from Statement 2, Lemma 3 and Statement 3. □

3. Towards infinite models with real time

First we give an informal description of a transition system with real time [3] which properties we want to verify.

Let S be a finite set of states. We will denote states by s, s', ... Let A be a set of actions which transfer system in the non-deterministic way from one state to another. In addition, to each action two constants from $\mathcal{N} \cup \{\infty\}$ are assigned; they form a duration interval for the action; we call them lower and upper time bounds. And for an action b with time bounds l and u we will write ${}_lb^u$ when necessary. The global variable t ranging over natural numbers will keep global system's time. For each action ${}_lb^u$ the variable T_b ranging over $\mathcal{N}$ will be called a timer of ${}_lb^u$ or a history variable. An individual timer can be incremented only simultaneously with the global time but it can independently be assigned to 0. An action is *enabled* in a state s iff the action may be executed in s. Let $En(b)$ be a set of all states where the action b is enabled. We say that an action ${}_lb^u$ is *eligible* iff $T_b \geq l$. An action *may occur* when it is enabled and eligible simultaneously. If an action may occur, its execution leads to transformation of a current state to a next one and its individual timer is assigned to 0. For any other action if it is enabled in the next state, its individual timer does not change its value; if not, the timer is assigned to 0. But there is one distinguished action ${}_0tick^\infty$ that differs from actions described above by the rules of transferring. *Tick* is enabled in every state and it does not change a state. However, it increments by 1 the global variable t and individual timers of all enabled actions. We will say that an action *must* (or *have to*) occur till next *tick* iff its individual timer is equal to the upper time bound of the action. The *tick* action is eligible iff there is no any action which must occur till next *tick*.

Thus, we obtained a transition system with actions transferring with time delay.

It is easy to understand that actions like ${}_0a^\infty$ do not need timers, because they are eligible just when they are enabled; we call such actions *untimed actions*; all the other actions are *timed actions*. Actions like ${}_\infty a^\infty$

do not have a sense at all, because they never occur. For actions like ${}_la^{\infty}$, where $l > 0$, the individual timers suffice reaching the lower time bounds and then maintaining it as long as the action is enabled, because the action will never actually have to occur before the next *tick*. At last, for actions like ${}_la^u$, where $0 < l \leq u < \infty$ the individual timers are bounded by the appropriate upper time bound. Therefore, we may redefine our system so that every timer has a finite range. Now we will describe formally a model for our transition system.

Definition 3. A configuration of a system is a vector given by

$$(s, h_1, \ldots, h_n),$$

where s is a state, $h_1, \ldots, h_n$ are values of timers of all timed actions a_1, $\ldots$, a_n.

Note, that the set of all configurations D_c is finite.

Definition 4. An extended configuration of a system is a vector given by

$$(s, h_1, \ldots, h_n, t),$$

where $(s, h_1, \ldots, h_n)$ is a configuration, and t is a value of the time variable t.

Note, that the set of all extended configurations D_t is countable.

It is clear that any status of the system with real time can be fully described by an extended configuration.

Definition 5. Let $M_t = (D_t, I_t)$ be a *real-time model*, where D_t is a set of all extended configurations, I_t is an interpretation of predicates and actions. We will say that I_t is *induced* by the transition system if the following holds:

- For any action ${}_la^u$ (including *tick*) and any $d_t, d'_{t'} \in D_t \quad (d_t, d'_{t'}) \in It(a)$ iff the action ${}_la^u$ may occur in the extended configuration d_t and it can transfer the system to the extended configuration $d'_{t'}$.

We consider also an *untimed* model $M_c = (D_c, I_c)$, where D_c is a set of all configurations, and I_c is a projection of the interpretation I_t on D_c. Formally, it means that I_c should meet two following conditions.

1. For any predicate $p \quad d \in I_c(p)$ iff there exists $t \in \mathcal{N}$ such that $d_t \in I_t(p)$;

2. For any action b $(d, d') \in I_c(b)$ iff there exist $t, t' \in N$ that $(d_t, d'_{t'}) \in I_t(b)$.

We will also say that the interpretation I_c is *induced* by the transition system if corresponding I_t is induced by this system. A model $M = (D_M, I_M)$ is *induced* by the transition system iff D_M is either the set of all configurations or the set of all extended configurations, and the interpretation I_M is induced by the transition system.

From now on we will fix some transition system and consider only models induced by this transition system.

If we could store and operate with countable sets with the help of computer, we would be able to evaluate formulae in the Mu-calculus with time using the model M_t and hence to verify any expressible property of a transition system with real time. In this paper we try to find classes of properties, for which we can reduce evaluating of appropriate formulae in the countable model M_t to the model-checking in some finite model. An accurate reader will note that the correctness of the semantics of formulae for some special infinite models is actually proved in Theorem 2, Corollary 1 and Theorem 3. Moreover, the semantics is proved to be correct for any countable model in [1].

The set of all extended configurations is infinite only because of infinity of range of variable t. The next theorem allows us to reduce verification of properties which do not depend on time, to model-checking in finite model M_c.

Theorem 2. *Let $M_t = (D_M, I_M)$ be a real-time model induced by a transition system. Let the interpretation I_t meet the following: for any predicate p $I_t(p) = I_c(p) \times \mathcal{N}$. Let also an evaluation $E_t : V \mapsto 2^{D_t}$ map the set of variables in the same way: $E_t(x) = E_c(x) \times \mathcal{N}$, where X is a variable and $E_c : V \mapsto 2^{D_c}$ is an evaluation in the model M_c. Then, for any formula F $[F, E_t]_t = [F, E_c]_c \times \mathcal{N}$.*

PROOF. The induction on the structure of F. Most cases are obvious; we consider here only two cases when F is $< a > F'$ and $max \quad x.F'$. The fact that $[< a > F', E_t]_t \subseteq [< a > F', E_c]_c \times \mathcal{N}$ follows from the constructing of M_c. To prove that $[< a > F', E_c]_c \times \mathcal{N} \subseteq [< a > F', E_t]_t$ it is sufficient to show that if $d_t \in [< a > F', E_t]_t$, then for any t' $d_{t'} \in [< a > F', E_t]_t$. By the definition of $[,]_t$, there exists $d'_{t'} \in [F', E_t]_t$ such that $(d_t, d'_{t'}) \in I_t(a)$. By the induction hypothesis, for any $t'' \in \mathcal{N}$ $d'_{t''} \in [F', E_t]_t$. Let us consider here only the case when $a = tick$. So, $d = d'$, $t' = t + 1$. Since for any $t'' \in \mathcal{N}$ $(d_{t''}, d_{t''+1}) \in I_t(tick)$, it follows that $d_{t''} \in [< a > F', E_t]_t$. Now

let us discuss the case of $max \quad x.F'$. We will compute the greatest fixed points of F' in two models M_c and M_t simultaneously.

$$x_0 = D_c, \qquad x_{t\ 0} = D_t = D_c \times \mathcal{N};$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$x_{i+1} = \left[F', E_c(x/x_i)\right]_c, \quad x_{t\ i+1} = \left[F', E_t(x/x_i)\right]_t$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

If for some i $x_{t\ i} = x_i \times \mathcal{N}$, then by the induction hypothesis, $x_{t\ i+1} = x_{i+1} \times \mathcal{N}$. Since D_c is finite there exists $n = |D_c|$ that $x_{n+1} = x_n$. Hence,

$$x_{t\ n+1} = x_{n+1} \times \mathcal{N} = x_n \times \mathcal{N} = x_{t\ n}\ .$$

Using Statement 1 we may conclude that $x_{t\ n}$ is the greatest fixed point of F', and

$$\left[max \quad x.F', E_t\right]_t = \left[max \quad x.F', E_c\right]_c \times \mathcal{N}.$$

□

Corollary 1. *Suppose, the interpretation in a model $M_t = (D_t, I_t)$ meets the following: for any predicate p if $d_t \in I_t(p)$ for some t, then $d_{t'} \in I_t(p)$ for any $t' \equiv t \pmod p$, where $p > 0$ is a fixed period. Let an evaluation E_t map the set of variables in the same way, i.e., if for some t $d_t \in E_t(x)$, then $d_{t'} \in E_t(x)$ for any $t' \equiv t \pmod p$, where x is a variable. Then for any formula F if some $d_t \in \left[F, E_t\right]_t$, then $d_{t'} \in \left[F, E_t\right]_t$ for any $t' \equiv t \pmod p$.*

SKETCH OF PROOF. We extend the set of actions by a new action ${}_p tact^p$. ${}_p tact^p$ is enabled in every state and does not change it when occurs. We also force ${}_p tact^p$ to occur as soon as possible, i.e., strait away after every p-th tick, so that the appropriate timer $T_{tact} \equiv t \pmod p$. Let model M_t be induced by this new transition system with *tact*. If the interpretation satisfies the conditions of Theorem 2 predicates and variables may depend on T_{tact}, so they may still depend on time indirectly and periodically as we need. To complete the proof one may use Theorem 2. □

Corollary 2 (The upper bound of complexity). *Evaluating of a formula in the model $M_t = (D_t, I_t)$ for a transition system with evaluation of predicates with a period p can be reduced to evaluating of the formula in the model $M' = (D', I')$, where $D' = D_c \times [0 \ldots p-1]$ and I' is the projection of I_t on D'.*

SKETCH OF PROOF. The model M' may be considered as a projection of M_t on D' when *tick* changes the variable t in the following way: $t :=$

$(t+1) \mod p$. Corollary 1 assures that one may expand the result of a formula's evaluating to a countable periodical set. □

To verify the properties like $t \leq k$ and $t \geq k$ it is sufficient to extend the model M' from Corollary 2 by the non-periodical range $[0 \ldots k-1]$ and to shift the periodical range from $[0 \ldots p-1]$ to $[k \ldots k+p-1]$. The expansion of this model to a countable set gives us semilinear time sets. The next theorem states the correctness of this expansion.

Definition 6. The semantic set $S_k^p \subseteq D_t$ is semilinear with the length of non-periodical part k and a period p if the following holds: If $d_t \in S_k^p$ and $t \geq k$, then for any $t' \equiv t \pmod p$, $t' \geq k : d_{t'} \in S_k^p$.

Theorem 3. *Let $M_t = (D_M, I_M)$ be a real-time model induced by a transition system. Let the interpretation I_t map the set of predicates to semilinear semantic sets. Let also an evaluation E_t map the set of variables to semilinear semantic sets. Then for any formula F $[F, E_t]_t$ is also a semilinear semantic set with the same period and non-periodical part.*

We omit here the proof because of its evidence.

Corollary (The upper bound of complexity). *Evaluating of a formula in the model $M_t = (D_t, I_t)$, induced by a transition system, with interpretation of predicates by semilinear semantic sets with a non-periodical part k and a period p can be reduced to evaluating of the formula in the model $M' = (D', I')$, where $D' = D_c \times [0 \ldots k+p-1]$ and I' is the projection of I_t on D'.*

SKETCH OF PROOF. Any semilinear semantic set can be represented by a subset of D'. Thus, we may construct the interpretation I' as a projection of I_t on D'. The correctness of expanding results of formula's evaluating in M' to a semilinear set follows from Teorem 3. □

Appendix
A brief description of the model-checker prototype

This section deals with a sketch of description of the prototype of a model-checker that is based on the polynomial model-checking procedure from Statement 3.

The prototype was implemented by Kerejbaev D.J. and Shnaider P.V. on IBM-PC AT 286 as a diploma project.

A development of the prototype makes it possible to improve some features of the algorithm and an expected system. For example, we developed a command language for parameterized sets of natural numbers, parameterized binary relations on natural numbers and some kind of simple recursion. Some non-efficiencies of the algorithm were also detected and eliminated along the implementation.

The prototype was implemented in a functional programming language REFAL. This choice is motivated by necessity of processing texts and fast implementation of a prototype. Unfortunately, some features of REFAL realization on IBM-PC restricts efficiency of the model checker prototype.

The prototype has a modular structure. The modular organization of the prototype reflects global steps of model-checking procedure: a model generation; the transformation of a formula to a system of equations and the evaluation of a system of equations.

Modules interact through special files, created by prototype during a work. Two entrance files must be created earlier. They are : the file of a model's description and the file of a formula.

Of course, the prototype supports some services: data files edition; lists of mistakes; activation of a module; etc. Interface is simple and reliable.

Acknowledgements. We would like to thank Aleksandr V. Rjazanov for kind attention and constructive advices about the draft version of our paper.

References

[1] R.S. Streett, E.A. Emerson, An automata theoretic decision procedure for the propositional Mu-calculus, Information and Control, Vol.81, No.3, 1989, 249-264.

[2] R. Cleaveland, M. Dreimueller and B. Steffen, Faster model-checker for the modal Mu-calculus, Proc. of TAV-91, Montreal, 383-400.

[3] J.S. Ostroff, Automated verificaion of timed transition models, Lect. Notes Comput. Sci., Vol.407, 1990, 247-256.