Verifying *L* specifications of reactive algorithms against temporal properties not expressible in the language *L*.

Anatoly Chebotarev
ancheb@gmail.com

One of the most important problems related to the development of reactive algorithms is to guarantee the correctness of the algorithm specification. This can be achieved only by formal verification of the desired properties of the algorithm. The verification consists in showing that the model of the algorithm to be verified satisfies correctness properties of its behavior. In order to be able to perform verification, one needs a formal model of the algorithm and a language for the formulation of properties. As a model of the algorithm, we consider its specification in quite simple logical language *L* [1]. Simplicity of this language enables to efficiently synthesize imperative representation of the algorithm from its declarative specification. Checking such specification for satisfiability plays an important role in the design process, so a rather high efficiency of a satisfiability checking algorithm is required. Efficient methods for checking *L* formulas for satisfiability has been developed [1], [2] and implemented in the corresponding tools. To formulate properties to be proved we use the General Reaction (1) class of LTL formulas [3]. Although the properties specified in this language are not expressible in the language *L*, we proposed an approach that reduces proving such properties to satisfiability checking *L* formulas with available tools for handling *L* specifications.

The language for specification of reactive algorithms.

The specification language *L* is a fragment of a first-order logic with monadic predicates interpreting over the set of integers **Z** (discrete time domain). A specification in the language *L* is of the form $\forall tF(t)$, where $F(t)$ is a formula constructed by means of logical connectives from atomic formulas (atoms) of the form $p(t + k)$, where p is a monadic predicate symbol, t is a variable ranging over **Z**, and k is an integer constant called the *rank of the atom*. Define the *depth of a formula* $F(t)$ to be the difference between the maximal and minimal ranks of the atoms occurring in $F(t)$. Since $F(t)$ is interpreted over the set of integers the equivalence $\forall tF(t) \Leftrightarrow \forall tF(t + k)$, where $F(t + k)$ denotes the formula obtained from $F(t)$ by adding k to the ranks of all its atoms, holds for any integer k. Hence, we may assume that the maximal rank of atoms occurring in any formula is equal to 0.

In defining the semantics of specification languages they are viewed as formalisms for describing sets of infinite words. Let Σ be a finite alphabet, **Z** the set of integers, and let $\mathbf{N}^+ = \{z \in \mathbf{Z} \mid z > 0\}$. Mappings u: $\mathbf{Z} \to \Sigma$ and l: $\mathbf{N}^+ \to \Sigma$ are called respectively a *bi-infinite word* (indicated as $\ldots u(-2)u(-1)u(0)u(1)u(2) \ldots$) and an ω-*word* (indicated as $l(1)l(2) \ldots$) over the alphabet Σ. The set of all bi-infinite words over Σ is denoted by Σ^Z. A segment $u(\tau)u(\tau+1) \ldots u(\tau+k)$ of a bi-infinite

word u is denoted by $u(\tau, \tau+k)$ and a bi-infinite word which is an infinite repetition of a nonempty word r, i.e. $r \cdot r \cdot r \ldots$, is denoted by r^Z.

The interpretation domain of the language L is **Z**, and only predicate symbols are uninterpreted. Thus an interpretation of the formula $\forall t F(t)$ is a collection of monadic predicates corresponding to all predicate symbols in Ω, i.e., q-tuple $<\pi_1, \pi_2, \ldots, \pi_q>$ of predicates. Each such predicate π_i we can regard as a bi-infinite word over the alphabet $\{0, 1\}$. A collection of q such predicates may be viewed as a bi-infinite word over the alphabet $\Sigma = \{0, 1\}^q$. We will not distinguish between interpretations and bi-infinite words over Σ and will speak of the truth value of the formula $F(t)$ at the position τ ($\tau \in$ **Z**) in the bi-infinite word u, to mean the evaluation of the formula $F(\tau)$ under the interpretation u. The meaning of the notion of the depth of a formula is that the truth value of the formula $F(t)$ of depth r at the position τ of the interpretation u is determined by the segment $u(\tau–r, \tau)$ of the corresponding bi-infinite word. The interpretation under which the formula $\forall t F(t)$ is true is called a *model* for this formula. Every formula $F = \forall t F(t)$ is associated with a set $M(F)$ of all models for this formula, i.e., bi-infinite words.

Verification of temporal properties not expressible in the language L.

To represent formulas specifying the algorithm and the property to be checked, we use the first order logic with monadic predicates. The algorithm specified by the formula $F = \forall t F(t)$ satisfies the property $\varphi = \forall t \varphi(t)$, if the formula $\forall t F(t) \rightarrow \forall t \varphi(t)$ is valid, i.e., all the models for F are models for φ. The properties which can be expressed in the language L, need not be proved, as they can be included in the algorithm specification, and the synthesized algorithm is guaranteed to satisfy these properties. The most simple property that is not expressible in L is of the form $\varphi = \forall t \exists t_1 (t_1 \geq t) F_1(t_1)$, where $F_1(t)$ is an L formula that contains no quantifiers. Specification F does not satisfy this property, if there exists its model u such that, beginning from a certain position τ of this model formula $\neg F_1(t)$ holds at every position $t \geq \tau$. Let the depth of formula $F(t)$ be r, i.e. the values of all its predicates at any position t of the interpretation u are determined by the segment $u(t–r, t)$ of the corresponding bi-infinite word. Then in the interpretation u there exist two positions τ_i and τ_j ($\tau < \tau_i < \tau_j$), such that $u(\tau_i–r, \tau_i) = u(\tau_j–r, \tau_j)$, and thus bi-infinite word $(u(\tau_i + 1, \tau_j))^Z$ is also a model for F. This model satisfies the property $\forall t \neg F_1(t)$. From the other hand, if there is a model satisfying this property then the algorithm specified by F does not satisfy the property φ. Thus specification F does not satisfy property φ if and only if the formula $\forall t F(t) \& \neg F_1(t)$ is satisfiable. Since this is an L formula, its satisfiability can be easily checked by the available tools based on the improved resolution method.

The property that has to be checked is often of the form $\varphi = \forall t(F_1(t) \rightarrow \exists t_1(t_1 \geq t)F_2(t_1))$, where $F_1(t)$ and $F_2(t_1)$ are L formulas that contain no quantifiers. First consider the case when $F_1(t)$ retains the value "true" till the time instant t_1. More precise this condition is expressed by the formula $\forall t(F_1(t)\&\neg F_2(t)\&\neg F_2(t+1) \rightarrow F_1(t+1))$. In this case the property φ can be replaced by the property $\varphi' = \forall t\exists t_1(t_1 \geq t)\&(\neg F_1(t_1) \vee F_2(t_1))$, which asserts that $F_1(t_1)\&\neg F_2(t)$ does not retain the value "true" for infinitely long time. Thus the problem reduces to the above case, i.e., to checking L formula $\forall tF(t)\&F_1(t)\&\neg F_2(t)$ for satisfiability.

We now consider the case when $F_1(t)$ does not retain the value "true" till the time instant t_1. In this case we add the formula $\varphi_z = \forall t(z(t) \leftrightarrow \exists t_1(t_1 \geq t)F_1(t_1)\&\neg F_2(t_1)\&\forall t_2((t_1 \leq t_2 \leq t) \rightarrow \neg F_2(t_2)))$ to the specification. Here, z is a new predicate symbol (not occurred in the specification). Now, the property to be proved can be expressed by the formula $\varphi' = \forall t\exists t_1(t_1 \geq t)\&\neg z(t_1)$, and proving this property amounts to checking the formula $\forall tF(t)\&\varphi_z(t)\&z(t)$ for satisfiability. Note that formula φ_z is not an L formula, therefore this specification has to be transformed into a specification in the language L. It has been shown that there exists a specification in the language L equivalent to this specification with respect to the semantics of the languages involved. This specification can be obtained by replacing formula φ_z by the L formula $\forall t(z(t) \leftrightarrow (z(t-1)\&\neg F_2(t) \vee F_1(t)\&\neg F_2(t)))$ and subsequent transformation of the resulting formula. The appropriate technique has been developed.

For open systems, the properties to be proved are, as a rule, of the form $\varphi^e \rightarrow \varphi^s$, where φ^e is a property of the environment and φ^s is a property of the algorithm to be verified. Formulas φ^e and φ^s are finite conjunctions of L formulas and formulas of the form $\forall t\exists t_1(t_1 \geq t)F_1(t_1)$ or $\forall t(F_1(t) \rightarrow \exists t_1(t_1 \geq t)\& F_2(t_1))$. Let $\forall tF_e(t)$ and $\forall tF(t)$ be L specifications of the environment and the algorithm, respectively. First we check whether the specification of the algorithm satisfies the property φ^s. As it has been shown above, such check can be performed by applying the resolution procedure to some formula of language L. If this check accomplishes successfully, we are done. If the check fails, which corresponds to satisfiability of the analyzed formula, we get a formula $\forall tF'(t)$ that characterize the set of all models for the specification, which do not satisfy the property φ^s. In this case, the system satisfies the property $\varphi^e \rightarrow \varphi^s$ if and only if the specification $\forall tF'(t)$ satisfies the property $\neg\varphi^e$. Let the property φ^e be of the form $\forall t(F_1(t) \rightarrow \exists t_1(t_1 \geq t)\& F_2(t_1))$. Without loss of generality, we may regard the property of the form $\forall t\exists t_1(t_1 \geq t)\&(\neg F_1(t_1) \vee F_2(t_1))$ instead of φ^e. Its negation is $\exists t\forall t_1((t_1 \geq t) \rightarrow F_1(t_1)\&\neg F_2(t_1))$. The specification does not satisfy this property if and only if there exist models that satisfy the property $\forall t\exists t_1(t_1 \geq t)\&(\neg F_1(t_1) \vee F_2(t_1))$. It can be easily shown that the existence of such models imply the existence of the models satisfying the property $\forall t\exists t_1(t_1 \leq t)\&(\neg F_1(t_1) \vee F_2(t_1))$. Hence, checking property $\varphi^e \rightarrow \varphi^s$ reduces to checking the formula $\forall tF'(t)\&\exists t_1(t_1 \leq t)\&(\neg F_1(t_1) \vee F_2(t_1))$ for satisfiability. To this end, the subformula $\exists t_1(t_1 \leq t)\&$

$(\neg F_1(t_1) \vee F_2(t_1))$ is replaced by the formula $z(t)\&(z(t) \leftrightarrow (z(t-1) \vee \neg F_1(t_1) \vee F_2(t_1)) = z(t)\&(z(t-1) \vee \neg F_1(t_1) \vee F_2(t_1))$, where z is a newly introduced predicate symbol. The resulting formula is transformed to obtain an L formula equivalent to $\forall t\, F'(t)\&\exists t_1(t_1 \le t)\&(\neg F_1(t_1) \vee F_2(t_1))$.

From the above discussion it follows that not only proving properties expressible in GR(1) language can be reduced to checking satisfiability of L formulas, but also more general kind of properties.

References

1. A. Chebotarev and M. Morokhovets. Consistency checking of automata function specifications. Proc. LPAR'93, Lecture Notes in Artificial Intelligence, v. 698, Springer, Berlin, 76 – 85, 1993.
2. A. Chebotarev, S. Krivoi. Improved Resolution-Based Method for Satisfiability Checking Formulas of the Language L. Proc. conf PSI'06, LNCS, vol. 4378, Springer, Berlin, 438 – 442, 2007.
3. N. Piterman, A. Pnueli, Y. Sa'ar. Synthesis of reactive(1) designs // Proc. of Conf, on Verification, Model Checking and Abstract Interpretation, 364–380, 2006.