Scenario based analysis of linear computations

Vladimir Filatov[1] and Rostislav Yavorskiy[2] ⋆

[1] Department of Mechanics and Mathematics
Moscow State University
Moscow, 119992, Russia
filatov@lpcs.math.msu.su

[2] Steklov Mathematical Institute
Gubkina 8, Moscow, 119991, Russia
rey@mi.ras.ru

Abstract. In this paper we consider the following task: given an abstract state machine, characterize the subsets of initial values corresponding to different typical scenarios of the system behavior. In order to solve it we suggest formalization of the notion of scenario and then discuss possible approaches to the classification of the computations.

1 Introduction

The research described here was motivated by our attempts to write in AsmL [1] and execute the model of a simple ecological system. The model describes dynamics of four populations according to laws formalized by a collection of linear equations. In other words, all the atomic updates are linear functions (a rather detailed description of the model is given in chapters 2, 3). When the model was completed, we have assigned some reasonable values to the parameters of the model (the coefficients) and provided some realistic initial values. The evolution terminated in 3 steps! We redefined the initial values — the same result. It took us some time to find initial values such that the computation would continue 20 steps or more. That was very surprising, because the model looked very natural and the coefficients were reasonable, therefore a priori we expected that randomly chosen initial values would give us an example of a very long scenario. Finally, we came to the following task, which seems to be interesting in general setting:

Given an abstract state machine (S, S_0, τ), characterize the subsets of initial values $S_0' \subseteq S_0$ corresponding to different typical scenarios of the system behavior.

In particular, it is interesting to investigate the problem for some specific classes of ASMs.

⋆ Partially supported by grants of President of Russia for Leading Scientific Schools, Russian Foundation for Basic Research, Russian Science Support Foundation, and Microsoft Research.

This general task has at least two different sides. First of all one has to define formally the notion of a scenario in this context. Then, the appropriate algorithms and technics should be developed.

The rest of the paper is organized as follows. In section 2 we provide some details on the motivating example. In section 3 all the necessary formal definitions are given and our approach to the problem given above is described. The conclusion is given in section 4.

2 Introductory example

In this section the model of the dynamics of four populations with discrete time is described.

2.1 The model

Consider the following ecological system. There are four populations: grass, rabbits, foxes and wolves. Wolves eat foxes and rabbits, foxes eat rabbits who eat grass. The state of our state transition system M is a structure that consists of a vector $\boldsymbol{x}$ in $\mathbb{R}^4$ and two natural numbers t_1, t_2 for time. The state space S can be defined in the following way:

$$S = \mathbb{R}^4 \times \mathbb{N}^2 \times \Omega$$

where Ω is a finite set needed to formalize the notion of a season and other finitely valued parameters.

We assume that the populations change according to the linear law. The transition function $\tau : S \to S$ is defined by the program that consists of the following lines:

$$\text{if } R_i \text{ then } \tau_i$$

Here $R_i, i = \overline{1, k}$ is a conjunction of simple conditions, τ_i is linear. The transition τ is deterministic.

We assume that the laws of interaction of animals change with seasons. In our model there are five conditions R_i which depend on $\boldsymbol{t} = (t_1, t_2) \in \mathbb{N}^2$ only. The first time parameter t_1 is the current step number while the second one t_2 defines the current day from the beginning of the year. Two updates correspond to the day of system life.

2.2 Variations

One can also consider a non-deterministic model. The function τ can be defined to be nondeterministic — the vector $\boldsymbol{x}$ is updated in the following way:

$$\boldsymbol{x} := (A_i + \Delta) \cdot \boldsymbol{x}$$

where Δ is a 4×4 matrix of random additions that cause nondeterminism.

Finally, the transition function (both the deterministic and the nondeterministic) can be $\tau : Input \times S \to S$ and can depend on external parameters from the set $Input$. $Input$ can be the same as Ω, it can be a set of matrixes Δ which define additions to A_i like in nondeterministic model and it can be a set of vectors $\boldsymbol{b}$, so that $\boldsymbol{x} := A_i \cdot \boldsymbol{x} + \boldsymbol{b}$. In such a way we can model the changes caused by different external factors and events.

3 The formal definitions

In this section we describe formally the general framework of linear computations. Besides ecosystems, one can think about controlling devices, P2P protocols etc. — many algorithms of different kind could be formalized by means of the linear ASMs.

3.1 The considered model of computation — linear ASMs

Let M be a state transition system (S, τ) where S is a state space, τ is a transition function. We are interested in the case when

$$S = \mathbb{R}^n \times \mathbb{Z}^m \times \Omega.$$

Here $\mathbb{R}$ stands for the field of reals, $\mathbb{Z}$ denote integers, Ω is a finite set. Practically, it means that the state of the studied system is completely characterized by a real-valued n-vector $\boldsymbol{x} \in \mathbb{R}^n$; an integer-valued m-vector $\boldsymbol{v} \in \mathbb{Z}^m$; and one variable of a finite range $w \in \Omega$.

A *simple linear update* has the following kind:

$$\begin{aligned}
&\boldsymbol{x} := A\boldsymbol{x} + \boldsymbol{a} \\
&\boldsymbol{v} := B\boldsymbol{v} + \boldsymbol{b} \\
&w := c.
\end{aligned}$$

Here, A is an $n \times n$ matrix with real coefficients, $\boldsymbol{a}$ is a constant vector from $\mathbb{R}^n$; B is an $m \times m$ matrix with integer coefficients, $\boldsymbol{b}$ is a constant vector from $\mathbb{Z}^m$; c is a constant from Ω.

A *simple condition* has one of the following forms:

$$\begin{aligned}
&\boldsymbol{a} \cdot \boldsymbol{x} = \alpha, \quad \boldsymbol{a} \cdot \boldsymbol{x} \neq \alpha, \quad \boldsymbol{a} \cdot \boldsymbol{x} < \alpha, \quad \boldsymbol{a} \cdot \boldsymbol{x} > \alpha; \\
&\boldsymbol{b} \cdot \boldsymbol{v} = \beta, \quad \boldsymbol{b} \cdot \boldsymbol{v} \neq \beta, \quad \boldsymbol{b} \cdot \boldsymbol{v} < \beta, \quad \boldsymbol{b} \cdot \boldsymbol{v} > \beta; \\
&w = c, \quad w \neq c.
\end{aligned}$$

Here the constants are such that $\boldsymbol{a} \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$, $\boldsymbol{b} \in \mathbb{Z}^m$, $\beta \in \mathbb{Z}$, $c \in \Omega$.

Now, a *transition* τ is defined by the following program

$$\begin{aligned}
&\text{if } R_1 \text{ then } \tau_1 \\
&\text{elseif } R_2 \text{ then } \tau_2 \\
&\qquad \ldots \\
&\text{elseif } R_k \text{ then } \tau_k \\
&\text{else } \tau_0
\end{aligned}$$

Here each R_i is a boolean combination of simple conditions, τ_i is a simple linear update. Without loss of generality one can assume that the boolean combination is always positive.

3.2 Predicate abstraction and scenarios

Suppose now that we are going to analyze a model from the class described above. In particular, we are going to study interesting behaviors of the system corresponding to different choice of the initial state.

First, let us fix the length l of the considered computations. In practice, one is usually interested in short scenarios $l < 50$ since they are appropriate for stepwise manual inspection. Another interesting value is $l \approx 1000$, this number is usually enough to grasp specific long-term properties of the system behavior. In our example $l = 1000$ corresponds to the period of one year and a half of the system life.

It is clear that there are infinitely many different computations even for the short size. So, we need to classify all the computations into several (not too many) classes.

We start with a well known idea of predicate abstraction.

Consider a set of predicates P_1, …, P_r over the state space S. One can define an equivalence relation on the set of all possible states of the system in the following way:

$$s_1 \approx s_2 \text{ iff } (P_1(s_1) \equiv P_1(s_2)) \wedge \cdots \wedge (P_r(s_1) \equiv P_r(s_2)).$$

Thus, instead of considering infinitely many states one can now look at the behavioral dynamics of the equivalence classes, so-called hyper-states. Obviously, the number of the hyper-states is at most 2^r, where r is the number of the predicates.

We can use this approach to define an equivalence relation on the set of all computations. Then, a scenario is an equivalence class of the computations.

The straightforward generalization would be the following. Consider two computations of the same length l:

$$\begin{aligned}\mathbf{s}' &= (s'_0, s'_1, \ldots, s'_l)\\ \mathbf{s}'' &= (s''_0, s''_1, \ldots, s''_l)\end{aligned}$$

We define them to be equal if for each step i the i-th states in the both computations belong to the same hyper-state. Namely,

$$\mathbf{s}' \approx \mathbf{s}'' \text{ iff } \forall i \in \{0, 1, \ldots, l\}(s'_i \approx s''_i).$$

The maximal possible number of different equivalence classes is $(2^r)^l$, where 2^r is the number of the hyper-states, l is the length of computations. This is not feasible even for small values of r and l. That is why we consider slightly different definition of a scenario.

The next definition is a generalization of the previous one in the following sense. If two computations are equivalent, $\mathbf{s}' \approx \mathbf{s}''$, then they should represent the same scenario.

Suppose, the set of the properties $P_1, \ldots, P_r$ is given. Let now φ be a formula in the language of temporal logic built on them, e.g. $\Box P_1$, $\Diamond(P_2 \rightarrow \bigcirc P_1)$ etc.

Taking into account our experience of the analysis of the concrete ecosystem, we come up with the following semiformal requirements on the definition of a scenario:

- Every scenario is characterized by a temporal formula.
- The formulas used should be quite simple and easy to understand by a human.
- The number of all possible scenarios should be feasible.
- The set of the scenarios should be complete, i.e. every computation corresponds to at least one scenario.

To make this definition formal we have to choose an appropriate set of the temporal formulas. We suggest to choose the following, below $i, j, k \in \{1 \ldots r\}$:

1. $\Box P_i,\ \Box\neg P_i,\ \Diamond P_i \wedge \Diamond\neg P_i$.
   Here, for every property P_i we consider three basic scenarios: the property is always true during a computation, the property never holds, and neither of these two is valid — the value is not stable.
2. $\Box(P_i \wedge P_j)$, $\Box(P_i \wedge \neg P_j)$, $\Box(\neg P_i \wedge P_j)$, $\Box(\neg P_i \wedge \neg P_j)$.
   In this group we try to catch a simple correlation between two different properties P_i and P_j — to find computations, where the both properties are stable. Note that in this case the scenario corresponding to the negation of a formula is in some sense expectable, and so is not so interesting. That is why we don't list here formula $\neg\Box(P_i \wedge P_j)$ etc.
3. $\Box(P_i \equiv P_j)$, $\Box(P_i \equiv \neg P_j)$;
   These formulas describe the case when the two properties change in synchronous way. Again, the negation is not interesting enough to be included.
4. $\Diamond(P_i \wedge P_j \wedge P_k)$, $\Diamond(\neg P_i \wedge P_j \wedge P_k)$, $\Diamond(P_i \wedge \neg P_j \wedge P_k)$, $\cdots$, $\Diamond(\neg P_i \wedge \neg P_j \wedge \neg P_k)$;
   The correlation between three different properties is an interesting issue, although, due to the complexity reasons, we check only whether a given combination of the three properties is reachable.

Suppose, we have r predicates. Then the first item gives us $3r$ formulas to be observed, the second group has $2r(r-1)$ formulas, the third one has $r(r-1)$ formulas, the fourth group includes $4r(r-1)(r-2)/3$ formulas. Thus we get the total of $(4r^3 - 9r^2 + 8r)/3$ formulas (scenarios). Compare with $(2^r)^l$ for the previous definition.

3.3 The suggested method

Now, the formal model is specified and the notion of scenario is formalized. The next step is to classify all the computations with respect to the considered set of scenarios.

To achieve this goal we, at first, intensively execute the model with randomly chosen initial values. For us, the target number is 10^6 random executions. For each computation we check satisfiability for all the temporal formulas from the list. Thus, we compute statistical frequency for every scenario (the computed value estimates probability of the scenario). For every scenario of non-zero frequency we keep an example computation that could be used later during the stage of manual analysis.

Eventually, on the base of these computational experiments we formulate several hypotheses of the kind "the scenario φ is impossible for the considered model".

Finally, due to the specific form of the transition function for linear ASMs we can rewrite every hypothesis as a Boolean combination of linear inequations. Then, standard out-of-the-shelf algorithms could be applied to prove the hypotheses.

4 Conclusion

In this paper we consider a specific class of Abstract State Machines, namely, linear ASMs. A motivating example and the formal definitions are given. To analyze the models we suggest a formalization of the notion of a typical scenario. The formalization uses r user defined Boolean properties of a state, and operators of temporal logic to describe specific interesting behaviors of the considered system. We suggest a list of $(4r^3 - 9r^2 + 8r)/3$ temporal formulas that, according to our experience, correspond to the interesting scenarios. To classify all the computations of the considered system in terms of the scenarios we use random execution and symbolic analysis of the program. On the stage of random execution some hypotheses about the properties of the system behavior are formulated. Then, domain specific algorithms are used for symbolic checking of these properties.

5 Acknowledgements

The authors are thankful to Yuri Gurevich, Anton Esin, Andrey Novikov and Nikita Mamardashvili for the fruitful discussion and numerous useful comments on the subject.

References

1. AsmL: The Abstract State Machine Language. Reference Manual. Modeled Computation LLC, 2002.
http://research.microsoft.com/fse/asml/
2. Egon Börger and Robert Stärk, **Abstract State Machines. A method for High-Level System Design and Analysis.** Springer-Verlag 2003.

3. Wolfgang Grieskamp, Yuri Gurevich, Wolfram Schulte, and Margus Veanes. *Generating Finite State Machines from Abstract State Machines.* In ISSTA 2002, International Symposium on Software Testing and Analysis, July 2002.
4. V. Volterra. *Lecons sur la Theorie Mathematique de la Lutte Pour La Vie,* Paris, 1931.
5. G. Yu. Riznichenko and A. B. Rubin. *Mathematical models of biological production processes.* Moscow State University, 1993 (in Russian).
6. O. F. Sadykov and I. E. Benenson. *Dynamics of a population of small mammals. Concepts, hypotheses, models.* Moscow, "Nauka", 1992 (in Russian).
7. P. Turchin. *Complex population dynamics: a theoretical/empirical synthesis.* Princenton and Oxford, 2003.
8. N. N. Bautin, E. A. Leontovich. *Methods and technics for quality analysis of planar dynamic systems.* Moscow, "Nauka", 1990 (in Russian).
9. A. N. Kolmogorov. *Study of mathematical models of populational dynamics.* Problems of Cybernetics, vol. 25, p. 100, 1972 (in Russian).
10. Jay W. Forrester. *World Dynamics,* Cambridge, Massachusetts Wright-Allen Press, Inc., 1971
11. G. G. Melinetskiy. *Chaos. Structures. Computational experiment. Introduction to non-linear dynamics.* 2002 (in Russian).
12. S. G. Pushkov. *Representation of dynamic systems in the state space: precise and approximate realization.* Barnaul, 2003 (in Russian).
13. Richard M. Crownover. *Introduction to Fractals and Chaos,* University of Missouri-Columbia, Jones and Barlett Publishers, Inc., 1995