Analysis and verification of navigation strategies by abstract interpretation of cellular automata

Gianluca Amato and Francesca Scozzari

Dipartimento di Economia
Università “G. d’Annunzio” di Chieti-Pescara – Italy
{amato,scozzari}@sci.unich.it

Abstract. We present a new approach to the analysis and verification of simple properties of character navigation. We model navigation strategies for virtual characters by cellular automata, and use standard abstract interpretation techniques for abstracting and verifying navigation properties.

Keywords: Static analysis, verification, navigation, abstract interpretation, cellular automata.

1 Introduction

Animated simulations and virtual worlds are often inhabited by characters which autonomously and intelligently move, without any external/human control. For instance, this is the case for background characters interacting with main characters. They need to navigate from one point to the other in the virtual land, in continuous motion, taking their decisions in real time, in according to the environment. When characters move in small groups (e.g. animals, soldiers), in addition to decide a navigation strategy (perceive objects and obstacles, find collision-free paths) they also need to obey to many other constraints (remain in group, avoid crowded areas).

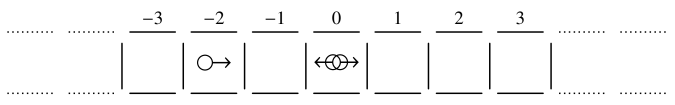
Analysis and verification of such properties in presence of multiple characters and obstacles heavily depend on the formalism used to describe the navigation strategy. For example, when using hybrid automata [2] as a model, it is possible to apply standard verification techniques to the navigation strategies for virtual characters. HyTech [16] is an example of a symbolic model checkers for linear hybrid systems, able to verify temporal properties [3]. When reasoning about navigation properties in animation system expressed in some temporal or modal logic for hybrid systems, we always need to introduce some degree of approximation or abstraction, in order to deal with the obvious undecidability problem of most properties. Moreover, in case of complex games, with many characters, groups of characters and a complex environment with obstacles, the verification problem may still be intractable, due to the high number of entities and constraints (see, e.g., [1]).

In order to overcome these problems, we suggest to model navigation strategies as cellular automata, and use abstract interpretation-based static analysis to prove run-time properties such as “no collision happens”.

2 Cellular automata

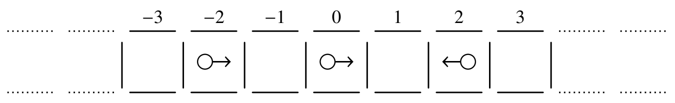
Cellular automata (CA) have been proposed to analyze the behavior of complex discrete dynamical systems with many interactions. A cellular automaton consists of a grid of cells, and each cell can be in a finite number of states (for instance: empty or occupied by specific entities). Interactions are usually described by a collection of simple rules. The time advances in discrete steps and the cellular automaton evolves according to its rules. Using cellular automata it is possible to model very complex structures, like ecological models, and easily simulate the interactions between a very large number of entities. Cellular automata have proved to be powerful enough to simulate thousands of interactions in real-time.

A simple example is that of a one-dimensional CA, where each cell can be in four different states: $\perp$ (empty), ○→ (right-directed), ←○ (left-directed) and ←○○→ (collision). Intuitively, the state ○→ can be thought of as a right-directed entity in a cell, and the state ←○○→ as a collision of two entities. For instance:

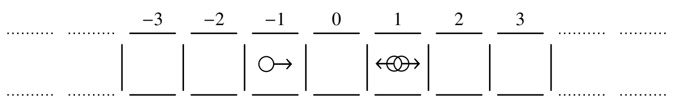


is a configuration where the cell in position −2 is in the state ○→ and the cell 0 is in the state ←○○→.

The rules are typically formalized by a function which computes the new state of a cell in terms of the current state of the cell and of some of its neighbors. The function for updating the state of a cell does not change over time and is applied to the whole grid simultaneously. Cellular automata can be equipped with many different types of rules, and the set of rules completely defines the behavior of the entities and obstacles involved in the game. For instance, applying obvious *moving* and *collision* rules to the configuration



we get the new configuration



3 Static analysis of cellular automata

The static analysis and verification of cellular automata amounts to fix some interesting properties and choose an approximation method to prove the properties. Here we propose to use the abstract interpretation theory to define a correct abstraction of CA, leveraging the big variety of abstract domains available for the analysis of symbolic and numerical properties. To this aim, we first need to embed cellular automata in the abstract interpretation framework [9].

3.1 Abstract interpretation

Abstract interpretation [12, 13] is a general theory for approximating the behavior of a discrete dynamic system. It discovers properties which hold when execution reaches specific program points. It can be used to observe many kinds of properties, e.g., termination, types and security properties, and it has been applied to many different fields, such as program analysis, verification and model checking. The key idea is to replace the (concrete) semantics of a system with an abstract semantics, computed over a domain of abstract objects. There are many different methods to describe the semantics of a system. Most of them are defined by a set of state-transition functions between states.

An abstract interpretation is specified by a set of *abstract objects*, called the *abstract domain*. The abstract objects describe the properties of the system we are interested in. The relationship between concrete and abstract objects is formalized by a pair of abstraction and concretization maps. The expressive power of abstract interpretation strictly depends on the particular choice of the abstract domain.

3.2 Abstract interpretation of cellular automata

We showed in [9] how to embed cellular automata in the abstract interpretation framework, which amounts to choosing a suitable concrete domain equipped with state-transition functions. The choice of the abstract domain depends on the properties we want to analyze. Simple properties of cellular automata are: “the cell *i* is empty”, “the cell *i* does not contain a right-directed entity”, “every cell whose index is odd does not contain a left-directed entity“. More realistic properties of navigation strategies, related to the description of regions in the game and to perceive obstacles, are naturally modeled by numerical abstract domains [15]. For instance, the properties of the kind: “a given region of cells is empty”, “the region contains a single entity”, or “the region only contains entities of a certain kind (enemies)” can be modeled as an interval of cell indexes in the one-dimensional case. Here we are abstracting the concept of “region”, which is a generic set of cells, by considering only regions made of consecutive cells. Generalization to many-dimensional cellular automata is easily realized using abstract domains such as convex polyhedra (similarly to PHAVer [14] for hybrid systems), octagons and parallelotopes. The power of these abstract domains is that they are endowed with polynomial operators (of very low degree), and many intractable problems may be over-approximated in a efficient and precise way. In particular, we suggest the use of template parallelotopes [5, 7, 10], a recently proposed abstract domain which combines dynamic and static analysis [11, 6]. The dynamic analysis gathers information about (partial) concrete executions, which is later exploited in the static phase. This allows to transfer the knowledge of the first part of the navigation strategy to tune the specific abstract domain (parallelotopes) to be used in the static phase.

In our early experiments, we have modeled and analyzed cellular automata with a finite (but unbounded) set of states and moving and collision rules. We have implemented a game with a finite set of characters and analyzed the property that a specific convex region does not contain any left-directed character. Using the observationally completeness technique [8, 4], we have found (an abstraction of) the set of the initial configurations for which the property holds. We plan to analyze more complex games, especially with many different characters and a large number of interactions, to fully exploit the power of the various numerical abstract domains when applied to the analysis of cellular automata. We believe that our approach could be specifically targeted to this kind of intractable analysis problems.

References

1. E. Aaron, F. Ivani, and D. Metaxas. Hybrid system models of navigation strategies for games and animations. In C. Tomlin and M. Greenstreet, editors, *Hybrid Systems: Computation and Control*, volume 2289 of *LNCS*, pages 22–84. Springer, 2002.
2. R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229, London, UK, 1993. Springer-Verlag.
3. R. Alur, T. A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. Softw. Eng.*, 22(3):181–201, 1996.
4. G. Amato, J. Lipton, and R. McGrail. On the algebraic structure of declarative programming languages. *Theoretical Computer Science*, 410(46):4626–4671, 2009.
5. G. Amato, M. Parton, and F. Scozzari. Deriving numerical abstract domains via principal component analysis. In R. Cousot and M. Martel, editors, *17th International Symposium, SAS 2010. Proceedings*, volume 6337 of *LNCS*, pages 134–150. Springer, 2010.
6. G. Amato, M. Parton, and F. Scozzari. A tool which mines partial execution traces to improve static analysis. In H. Barringer and *et al.*, editors, *First International Conference, RV 2010. Proceedings*, volume 6418 of *LNCS*, pages 475–479. Springer, 2010.
7. G. Amato, M. Parton, and F. Scozzari. Discovering invariants via simple component analysis. *Journal of Symbolic Computation*, 47(12), 2012.
8. G. Amato and F. Scozzari. Observational completeness on abstract interpretation. In H. Ono, M. Kanazawa, and R. de Queiroz, editors, *Logic, Language, Information and Computation, 16th International Workshop, WoLLIC 2009*, volume 5514 of *LNCS*, pages 99–112. Springer, 2009.
9. G. Amato and F. Scozzari. Observational completeness on abstract interpretation. *Fundamenta Informaticae*, 106(2–4):149–173, 2011.
10. G. Amato and F. Scozzari. The abstract domain of parallelotopes. In J. Midtgaard and M. Might, editors, *The Fourth International Workshop on Numerical and Symbolic Abstract Domains (NSAD 2012)*, Electronic Notes in Theoretical Computer Science. Elsevier, 2012.
11. G. Amato and F. Scozzari. Random: R-based Analyzer for Numerical DOMains. In N. Bjrner and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning 18th International Conference, LPAR-18, 2012. Proceedings*, volume 7180 of *LNCS*, pages 375–382. Springer, 2012.
12. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *POPL '79: Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 269–282. ACM Press, New York, 1979.
13. P. Cousot and R. Cousot. Abstract interpretation and applications to logic programs. *The Journal of Logic Programming*, 13(2–3):103–179, 1992.
14. G. Frehse. Phaver: algorithmic verification of hybrid systems past hytech. *Int. J. Softw. Tools Technol. Transf.*, 10(3):263–279, 2008.
15. T. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond hytech: Hybrid systems analysis using interval numerical methods. In N. Lynch and B. Krogh, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *LNCS*, pages 130–144. Springer, 2000.
16. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: a model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1:110–122, 1997.