Extracting a Winning Strategy in a Game: A Case Study of Theseus and Minotaur'

Ryoungkwo Kwon, Gihwon Kwon

Department of Computer Science, Kyonggi University,
Suwon-si, Gyeonggi-do, South Korea
{rkay8496, khkwon} @kyonggi.ac.kr

Abstract. In this paper, we apply GR(1) synthesis to extract a winning strategy for Theseus and Minotaur which is a two-player concurrent game. We first give an introduction of GR(1) synthesis, and then formalize Theseus and Minotaur with respect to GR(1). Finally, we show our experimental results.

Keywords: GR(1) Synthesis, Winning Strategy, Game Solution

1 Introduction

In this paper, we apply GR(1) - Generalized Reactivity(1) - synthesis to extract a winning strategy for Theseus and Minotaur which is a two-player concurrent game. Theoretically, GR(1) synthesis can be regarded as a game between an environment and a system[1]. Assume that there is a specification in LTL (Linear Temporal Logic). If a specification is realizable which means that a system wins no matter what an environment does, then GR(1) synthesis gives a winning strategy for the system[2]. The winning strategy is regarded as a solution for Theseus and Minotaur. With this technique, we could solve all the games from Level 1 through Level 87, which is the most difficult one in our context.

This paper is organized as follow. In section 2, we introduce some preliminaries which is related with GR(1) synthesis. In section 3, we explain our formulation of Theseus and Minotaur and then we apply our formulation to the game and show our experimental results in section 4. Finally, we give conclusions and future works in section 5.

2 GR(1) Synthesis

The GR(1) synthesis can obtain an implementation from a specification, if the winning strategy exist in the specification written by logical formula. Otherwise, if

I This work was supported by the GRRC program of Gyeonggi province. [2012, Development of Contents Application Software Technologies Based on Mobile Platforms.]

not, we called unrealizable, then the winning strategy does not exist in the specifications so that we can't obtain an implementation.

The GR(1) synthesis is considered as a two-player game between an environment and a system. Each player has a goal to win the game. A system win the game, if the system can be infinitely often reached a winning condition. Otherwise, the environment wins the game because of spoiling the system to infinitely often reach a winning condition. The winning condition is formed as generalized reactivity(1) formula(1).

$$(1110p1A—AlilOpm)(1110q1A...A1110qn) \qquad (^{1})$$

The p_m characterizes assumptions about the environment, and the q_n characterizes requirements about the system. So, the above condition means that the system wins the game if the environment infinitely often satisfies assumptions, and the system also infinitely often reaches the requirements in every run of the game.

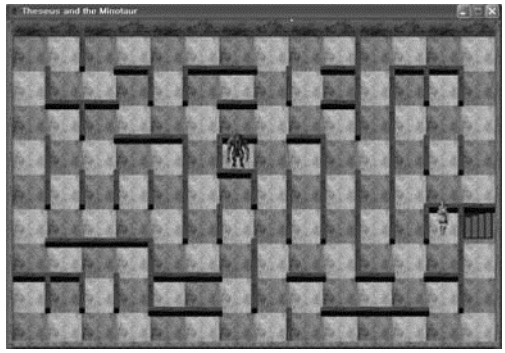
The game is played on a two player game structure which is formally defined as a tuple, Game Structure G=(*V, X,* Y, *0, p_e, p_s,* consisting of the following components. V represents a finite set of state variables over a finite domain, *X* is the finite set of variables which is controlled by the environment, and Y is the finite set of variables which is controlled by the system. O is the initial condition characterizing initial state of the game structure, *pe* and p_s represents transition relations of the environment and system respectively. Finally, cp is a winning condition.

A strategy is a partial function, *f.* V^fxXi-Y, which is mapping a series of states to possible system's next action. A run of the game is compliant with the strategy *f,* if each system's action is taken by the strategy *f* The strategy *f* is the winning strategy for the system, if every run where are compliant with the strategy f, win for the system. Thus, we can compute a set of winning states, W, where exists the winning strategy for the system. Otherwise, we can dually compute W_e.

The winning condition $_{cP}$ in the game structure is realizable, if initial states satisfying initial condition of the game structure are included in W. So not, $_{cP}$ is not realizable.

3 Problem Formulation

Theseus and Minotaur is a 2-player game between Theseus and Minotaur in Fig.1 . Theseus has to escape each maze by moving exactly one square in own turn, before the Minotaur catches you. Minotaur is twice as fast as Theseus is, but luckily, he moves in a predictable pattern. First Minotaur tests if he can move horizontally and get closer to Theseus. If he can, he will move one square horizontally. Otherwise, he will test if he can move vertically and get closer to Theseus. If he can, he will move one square vertically. If he can't move either horizontally or vertically, then he just skips that turn. We begin by introducing the variables used in our formulations and then describe the transition relation, initial conditions, and winning conditions for each player.



101	102	103	104	105	106	107	108	109	110	111	112	113	114
201	202	203	204	205	206	207	208	209	210	211	212	213	214
301	302	303	304	305	306	307	308	309	310	311	312	313	314
401	402	403	404	405	406	407	408	409	410	411	412	413	414
501	502	503	504	505	506	507	508	509	510	511	512	513	514
601	602	603	604	605	606	607	608	609	610	611	612	613	614
701	702	703	704	705	706	707	708	709	710	711	712	713	714
801	802	803	804	805	806	807	808	809	810	811	812	813	814
901	902	903	904	905	906	907	908	909	910	911	912	913	914

Fig. 1. The game and its abstracted positions

3.1 Variables

We formulate a game structure with a set of variables controlled by environment as inputs, and variables controlled by system as outputs. The inputs are as follows. S. is game spaces for the Minotaur. The value ranges over 3-digit numbers exceptionally including 0. That is, the value is currently 102 if the Minotaur is located at row 1, and column 2.

The outputs are defined as follow.

- S_t is game spaces for the Theseus. The value ranges over 3-digit numbers exceptionally including 999.
- *Count* is a Boolean variable. The count is false when Theseus has own turn. Otherwise, the count is true.

3.2 Transition Relations

We express all possible transitions relations for both players, Minotaur(environment) and system(Theseus), as a SMV format.

For transitions of an environment, we first express behaviors of the Minotaur. As described earlier, Minotaur move two square if possible. But Minotaur move in predictable pattern. Thus, we express Minotaur's behaviors, whether the Minotaur is left or right on the Theseus, by using mod operator in SMV.

Next, we express behaviors of the Theseus. The Theseus move one square in own turn. That is, S_t's value will change +1, -1, +100, or -100 in each turn possibly. Also Theseus can't move through a wall. The Theseus only takes a turn to move when the count is false. So, the Theseus has to stay current position, when the count is true.

The Theseus did not catch from the Minotaur before Theseus reaches a goal position. So, we can express that both player's position is not equal before Theseus reach the goal position.

3.3 **Initial and winning conditions**

The initial condition of our game structure is to express initial position of Minotaur and Theseus, and value of count variable. Exceptionally, an initial position of Minotaur is 0, because environment is first player on GR(1) synthesis, so environment will move initial position at first step. Thus, we can express that system is first player on our formalism. The initial conditions are as follows when the initial position of Theseus is 613.

$$(S_m=407AS,\text{--}613ACount) \quad (2)$$

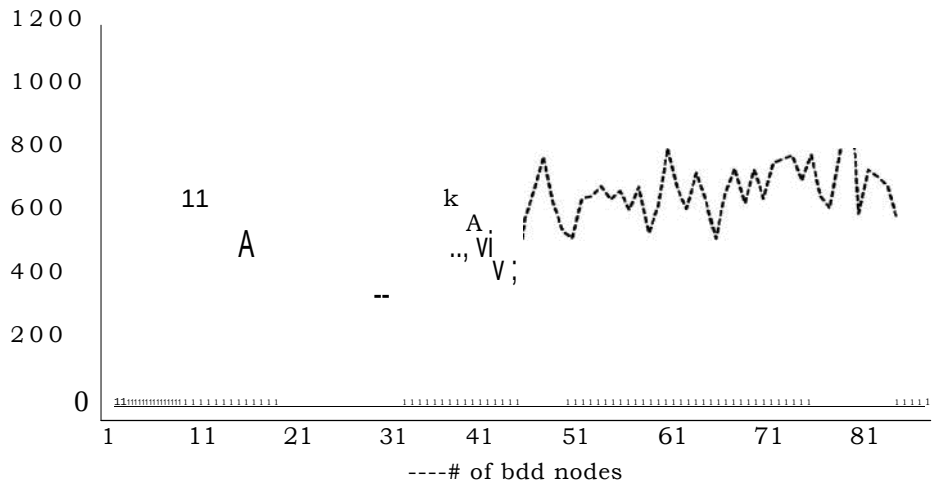
The winning conditions of both player are generalized reactivity(1) formula as follow when the goal position of Theseus is 614.

$$(\text{III } 0 \text{ TRUEA1110 } S_t=614) \quad (^3)$$

The goal of an environment is infinitely often true because no Minotaur has winning conditions. But, the Theseus has a goal.

4 Experimental Results

We experiment all stages of Theseus and the Minotaur, Kristanix[3]. We formulate the each stage by our formulation and then use JTLV[4] for GR(1) synthesis algorithm calculating winning states. Fig. 2 shows graphs of bdd nodes and game time of calculating winning states for the Theseus and Minotaur game. In these graphs, we derive a fact that bdd nodes and game time is gradually increment.



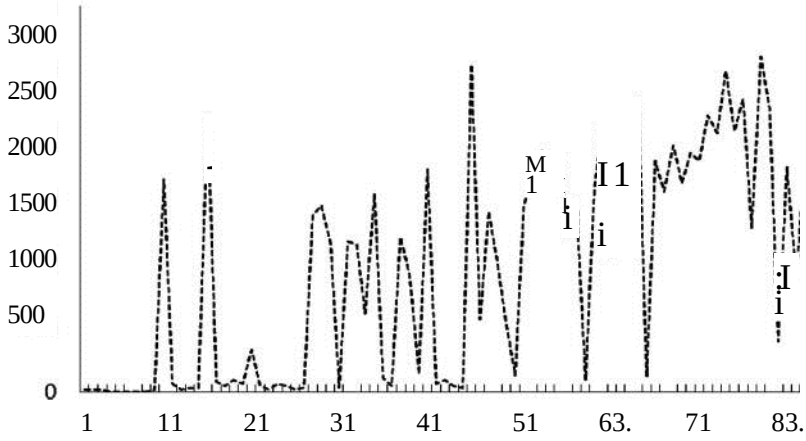


Fig. 2. The bdd nodes and game times for calculating winning states.

5 Conclusions and Future Works

In this paper, we present a method to extract a winning strategy by using the GR(1) synthesis. We have introduced what GR(1) synthesis is, especially including generalized reactivity(1) formula as a winning conditions. And we have explained how to express the Theseus and Minotaur game as a game structure formally defined for the GR(1) synthesis. Finally, we have modeled all stages of this game in our formulation and solved successfully.

We are currently trying to apply our experiences through this paper to other applications, including hardware/software design or automatic code generation because we believe that our experiences through this paper are useful.

References

1. N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of Reactive(1) Designs, In Proc. 7th Intl Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI'06), LNCS vol. 3855, pp. 364--380, 2006.
2. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Proc. 16th ACM Symp. Princ. of Prog. Lang., pp. 179-190 (1989)
3. http://www.kristanix.com/TheseusandtheMinotaur/
4. N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of Reactive(1) Designs, In Proc. 7th Intl Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI'06), LNCS vol. 3855, pp. 364--380, 2006.