Untitled CAV Paper

Alexander Legg[1], Leonid Ryzhyk[2], and Nina Narodytska[2]

[1] NICTA* and UNSW
alexander.legg@nicta.com.au
[2] Samsung Research

Abstract. This is the abstract

1 Introduction

Reactive systems are ubiquitous in real-world problems such as circuit design, industrial automation, or device drivers. Automatic synthesis can provide a *correct by construction* controller for a reactive system from a specification. However, the reactive synthesis problem is 2EXPTIME-complete so naive algorithms are infeasible on even simple systems.

Reactive synthesis is formalised as a game between the *controller* and its *environment*. In this work we focus on safety games, in which the controller must prevent the environment from forcing the game into an error state. Much of the complexity of reactive synthesis stems from tracking the set of states in which a player is winning.

There are several techniques that aim to mitigate this complexity by representing states symbolically. Historically the most successful technique has been to use *Binary Decision Diagrams* (BDDs). BDDs efficiently represent a relation on a set of game variables but in the worst case the representation may be exponential. This means that BDDs are not a one-size-fits-all solution for all reactive synthesis specifications.

Advances in SAT solving technology has prompted research into its applicability to synthesis as an alternative to BDDs. One approach is to find sets of states in CNF [**?**]. Another approach is to eschew states and focus on *runs* of the game. Previous work has applied this idea to realizability of bounded games [**?**] by forming abstract representations of the game. In this paper, we extend this idea to unbounded games by constructing approximate sets of winning states from abstract trees.

2 Reactive Synthesis

A *safety game*, $G = \langle X, L_c, L_u, \delta, I, E, \rangle$, consists of a set of state variables, sets of controllable and uncontrollable label variables, a transition relationship

* NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

$\delta : (X, L_c, L_u) \to X$, an initial state, and an error state. The *controller* and *environment* players choose controllable and uncontrollable labels respectively and the game proceeds according to δ.

An *run* of a game $(x_0, c_0, u_0), (x_1, c_1, u_1) \dots (x_n, c_n, u_n)$ is a chain of state and label pairs of length n s.t. $x_{k+1} \leftarrow \delta(x_k, c_k, u_k)$. A run is winning for the controller if $x_0 = I \wedge \forall i \in \{1..n\}(x_i \neq E)$. In a bounded game of rank n all runs are restricted to length n, whereas unbounded games consider runs of infinite length.

A *controller strategy* $\pi^c : X \to L_c$ is a mapping of states to controllable labels. A controller strategy is winning in a bounded game of rank n if all runs $(x_0, \pi^c(x_0), u_0), (x_1, \pi^c(x_1), u_1) \dots (x_n, \pi^c(x_n), u_n)$ are winning. Bounded *realizability* is the problem of determining the existence of such a strategy for a bounded game.

An *environment strategy* $\pi^e : (X, L_c) \to L_u$ is a mapping of states and controllable labels to uncontrollable labels. A bounded run is winning for the environment if $x_0 = I \wedge \exists i \in \{1..n\}(x_i = E)$ and an environment strategy is winning for a bounded game if there exists a run $(x_0, c_1, \pi^e(x_1, c_1)), (x_1, c_1, \pi^e(x_1, c_1)) \dots (x_n, c_n, \pi^e(x_n, c_n))$ that wins for the environment. Safety games are zero sum, therefore the existence of a controller strategy implies the nonexistence of an environment strategy and vice versa.

A set of runs for a bounded game of rank n can be represented by a *game tree* of depth n with states in the nodes and labels as edges. A *strategy tree* similarly represents a strategy by restricting the edges in a game tree to obey π^c for a controller strategy or π^e for an environment strategy.

2.1 Abstract Game Trees

The bounded synthesis algorithm from [**?**] constructs abstractions of the game in the form of *abstract game trees*.