# Generating Dominant Strategies for Continuous Two-Player Zero-Sum Games

**Marcell J. Vazquez-Chanlatte** *, **Shromona Ghosh** *, **Vasumathi Raman** **, **Alberto Sangiovanni-Vincentelli** *, **Sanjit A. Seshia** *

* *University of California, Berkeley, Berkeley, CA (e-mail: {marcell.vc,shromona.ghosh,alberto,sseshia}@eecs.berkeley.edu).*
** *San Francisco, CA (e-mail: vasumathi.raman@gmail.com)*

**Abstract:** Motivated by the synthesis of controllers from high-level temporal specifications, we present two algorithms to compute dominant strategies for continuous two-player zero-sum games based on the Counter-Example Guided Inductive Synthesis (CEGIS) paradigm. In CEGIS, we iteratively propose candidate dominant strategies and find counterexamples. For scalability, past work has constrained the number of counterexamples used to generate new candidates, which leads to oscillations and incompleteness, even in very simple examples. The first algorithm combines Satisfiability Modulo Theory (SMT) solving with optimization to efficiently implement CEGIS. The second abstracts previously refuted strategies, while maintaining a finite counterexample set. We characterize sufficient conditions for soundness and termination, and show that both algorithms are sound and terminate. Additionally, the second approach can be made complete to within an arbitrary factor. We conclude by comparing across different variants of CEGIS.

*Keywords:* Controller Synthesis, Robust Control, Counter-Example Guided Inductive Synthesis

## 1. INTRODUCTION

Correct-by-construction controller synthesis from high-level formal specifications offers a promising means of raising the level of abstraction for implementation. In particular, *reactive synthesis* from temporal logic generates programs or controllers which maintain an ongoing interaction with their (possibly adversarial) environments. Reactive synthesis from linear temporal logic using automata-theoretic methods has been demonstrated for synthesizing high-level controllers for robotics. However, for embedded, cyber-physical systems (CPS), reactive synthesis becomes much more challenging for several reasons. First, the specification languages go from discrete-time, propositional temporal logics to metric-time temporal logics over both continuous and discrete signals, so the previous automata-theoretic methods do not easily extend. Second, even for simple classes of dynamical systems and metric-time temporal logics, verification is itself undecidable, let alone synthesis. Third, the state of the art for solving games over infinite state spaces, as required for metric or quantitative temporal objectives, is far less developed than that for finite games.

To address these challenges, researchers have resorted to various simplifications. One simplification is to consider the control problem over a finite horizon instead of infinite horizons. This ensures verification is decidable for many interesting and practical systems and specification languages, like metric temporal logic (MTL) (Koymans, 1990) and signal temporal logic (STL) (Maler and Nickovic, 2004). Reachability (Mitchell et al., 2005) based techniques offer a suite of well-developed and mature tools for handling finite horizon games, whose constraints amount to simple safety invariants such as obstacle avoidance. However, direct extensions to temporal constraints are not straightforward. Existing approaches model the environment as a bounded, non-deterministic disturbance (Ding et al., 2011), which is often unrealistic, leading to infeasibility in applications like autonomous driving for even very mundane cases. Instead, one may have a more complicated model of the environment that leverages either data-driven techniques or known aspects of the behavior of the other agents, such as their formal specifications. Our contributions are motivated by improving the ability to use such models to constrain the environment in non-trivial ways.

A promising and scalable approach is to formulate reactive synthesis over a finite, receding horizon as a series of zero-sum games between the system and the environment, where the environment assumptions and system objectives are systematically encoded into rewards (Raman et al., 2015). This form of controller, known as Receding Horizon Control (RHC), offers a (limited) form of reactivity. Each game is solved using a *Counter-Example Guided Inductive Synthesis* (CEGIS) (Solar-Lezama et al., 2006) scheme to search for a dominant strategy. In particular, one first fixes a system strategy and encodes the environment's play as the solution to an Optimization Problem (OP). If the system's reward is negative, the environment's play is called a counterexample. The system then proposes a new candidate strategy that simultaneously considers all previous counterexamples. This process iterates until a dominant strategy is found. This framework has seen success on (hybrid-)linear dynamical systems and specifications in a large fragment of Signal Temporal Logic, where the resulting OP reduces to a Mixed Integer Linear Programming (MILP) Problem (Raman et al., 2015).

Practical implementations of CEGIS suffers from two major shortcomings. First, CEGIS is an iterative paradigm that produces a series of candidate solutions with corresponding counterexamples. In each iteration of CEGIS, one must simultaneously solve the original OP for all previously found counterexamples. In this paper, we refer to this ideal variant of CEGIS as $\mathcal{C}_N$. The authors in (Raman et al., 2015) speculated that in the case of MILPs, $\mathcal{C}_N$ would scale poorly as the set of counterexamples grew; but prior to this paper, this had not been empirically demonstrated. Nevertheless,

this scaling issue was circumvented by considering only the most recent counterexample in each iteration of CEGIS ($\mathcal{C}_{SCE}$). This heuristic leads to the second shortcoming, namely, that $\mathcal{C}_{SCE}$ may oscillate indefinitely between the same set of counterexamples.

*Example 1.* Consider the situation shown in Fig 1. Depending on the ego car's behavior, the lead car may move to block the ego car, leading to a crash. If the ego car stops, it may be rear ended. Reasoning about the infeasibility of reaching the goal safely under an adversarial environment requires access to at least three counterexamples. Maintaining only the last counterexample will cause the CEGIS to loop indefinitely. An ideal synthesis procedure would quickly reveal that the environment assumptions need to be strengthened (e.g. assume that given enough time the other cars would avoid the ego car or stop). Such discrete sub-problems arise naturally within more complicated (continuous space and time) driving scenarios.

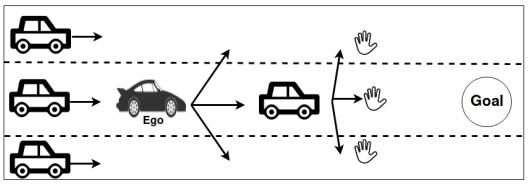


Fig. 1. The ego car is attempting to reach the goal while avoiding collisions with the other cars.

To deploy a RHC on a complex CPS, the synthesis procedure associated with each planning step must be implemented in real time, and making CEGIS practical is an important first step. However, prior efforts to speed up CEGIS lead to incompleteness or non-termination. For example, (Farahani et al., 2015) evaluated alternate techniques such as Monte Carlo and dual formulations for CEGIS to improve tractability. These techniques either provide only probabilistic completeness, or suffer from similar scaling-related issues.

In this paper, we present techniques and assumptions under which one can make the CEGIS step of receding horizon reactive synthesis practical, and characterize its termination in a fairly general setting. We believe this marks an important step towards synthesizing reactive controllers in real time. Our contributions are as follows:

(1) Empirical results verifying that, as hypothesized in (Raman et al., 2015), naively incorporating new counterexamples in $\mathcal{C}_N$ quickly becomes infeasible.
(2) A variant of $\mathcal{C}_{SCE}$, called $\mathcal{C}_{HYB}$, that adds fewer constraints per iteration, avoids oscillations, and outperforms $\mathcal{C}_N$ in certain circumstances.
(3) A variant of $\mathcal{C}_N$, called $\mathcal{C}_{SN}$, that proposes dominant feasible strategies rather than optimal robust strategies. We demonstrate that $\mathcal{C}_{SN}$ outperforms $\mathcal{C}_N, \mathcal{C}_{SCE}$, and $\mathcal{C}_{HYB}$. This comes at the cost of not yielding an "optimal" robust strategy, where optimality is with respect to a reward function capturing quantitative satisfaction of the specification.
(4) A worst-case convergence characterization for $\mathcal{C}_N$, $\mathcal{C}_{HYB}$, and $\mathcal{C}_{SN}$ which can be used for termination thresholds. We additionally show that $\mathcal{C}_{HYB}$ finds an $\epsilon$-robust solution when one exists, and provide an $\epsilon$-completeness proof.

## 2. PROBLEM STATEMENT

We seek solutions to the query:

$$\exists \mathbf{u} \in \mathcal{U} \ . \ \forall \mathbf{w} \in \mathcal{W} \ . \ \phi(\mathbf{u}, \mathbf{w}) \tag{1}$$

| Symbol | Optimal | Memory | "Terminates" | New |
|---|---|---|---|---|
| $\mathcal{C}_N$ | Yes | $\infty$ | Yes | No |
| $\mathcal{C}_{SCE}$ | Yes | 1 | No | No |
| $\mathcal{C}_{HYB}$ | Yes | $k \in \mathbb{N}$ | Yes | Yes |
| $\mathcal{C}_{SN}$ | No | $\infty$ | Yes | Yes |

Table 1. Summarizes the algorithms studied in this paper. "Optimal" means maximal with respect to a reward function capturing satisfaction of the high-level specification. "Memory" refers to the number of counterexamples the candidate oracle takes into account before proposing a candidate strategy. "Terminates" denotes whether the algorithm terminates.

where $\mathbf{u}$ ($\mathbf{w}$) encodes player 1's (2's) move, $\phi : \mathcal{U} \times \mathcal{W} \to \{True, False\}$ encodes player 1's objective and $\neg\phi$ is player 2's objective. Both $\mathcal{U}$ and $\mathcal{W}$ are assumed to be subsets of $[-1,1]^{n_u}$ and $[-1,1]^{n_w}$ respectively, where $n_u$ ($n_w$) are the dimensions of $\mathcal{U}$ ($\mathcal{W}$). As is common in the literature, we will often call player 1 the system and player 2 the environment. Thus (1) queries whether the system has a dominant strategy. [1]

*Remark 1.* In control applications, $\mathbf{u}$ and $\mathbf{w}$ are the sequence of control inputs for the system and environment response, and $\phi$ is typically a temporal constraint such as: "Eventually reach location A and always maintain at least 2 meters from all other objects".

In addition to $\phi$, we assume access to a function $\rho : \mathcal{U} \times \mathcal{W} \to \mathbb{R}$, called the quantitative semantics for $\phi$, such that:

$$\begin{aligned} \rho(\mathbf{u}, \mathbf{w}) > 0 &\implies \phi(\mathbf{u}, \mathbf{w}) \textit{ is True} \\ \rho(\mathbf{u}, \mathbf{w}) < 0 &\implies \phi(\mathbf{u}, \mathbf{w}) \textit{ is False} \end{aligned} \tag{3}$$

and $\rho$ is Lipschitz continuous in $\mathbf{u}$. That is, there exists a constant $L_\rho$ such that for all disturbances $\mathbf{w} \in \mathcal{W}$:

$$\forall \mathbf{u}, \mathbf{u}' \in \mathcal{U} \ . \ |\rho(\mathbf{u}, \mathbf{w}) - \rho(\mathbf{u}', \mathbf{w})| \leq L_\rho |\mathbf{u} - \mathbf{u}'| \tag{4}$$

*Remark 2.* The function $\rho(\mathbf{u}, \mathbf{w})$ is often interpreted as a quantitative measure of satisfaction of $\phi(\mathbf{u}, \mathbf{w})$. Larger values offer a higher degree of satisfaction and lower values offer a lower degree of satisfaction. An $\epsilon$-robust solution is a $\mathbf{u}$ such that $\forall \mathbf{w} \in \mathcal{W} \ . \ \rho(\mathbf{u}, \mathbf{w}) > \epsilon$. Such solutions are often desirable in RHC as they heuristically offer more resilience to the uncertainty introduced by modeling errors and the finite horizon approximations.

*Example 2.* $(\mathbf{u}, \mathbf{w}) \overset{\rho}{\mapsto} 0$ trivially satisfies (3) and (4).

*Example 3.* Consider the specification "For the next 30 seconds, $x > 2$ implies that $y$ will be less than 4 within two seconds" on a system $x(t+1) = x(t) + u(t) + w(t)$ and $y(t+1) = y(t) + u(t) + w(t)$ for bounded $u(t)$, $w(t)$. Consider the following syntactically- generated quantitative semantics over the state, following the quantitative semantics for STL defined in (Maler and Nickovic, 2004).

$$(x, y) \overset{r}{\mapsto} \max_{t \in \{0,\ldots,30\}} \left( x(t) - 2, \min_{t' \in \{t,\ldots,t+3\}} (4 - y(t')) \right) \tag{5}$$

Since $u(t)$ and $w(t)$ are both bounded and $r$ is smooth, substituting the dynamics equations into (5) to get a function over $u(t)$, $w(t)$ yields a quantitative semantics satisfying (3) and (4).

[1] Observe that one can take $\phi(\mathbf{u}, \mathbf{w})$ to be of the form:

$$\phi(\mathbf{u}, \mathbf{w}) \triangleq (\psi^A(\mathbf{u}, \mathbf{w}) \implies \psi^G(\mathbf{u}, \mathbf{w})) \tag{2}$$

where $\psi^A(\mathbf{u}, \mathbf{w})$ encodes assumptions (or constraints) on the actions of the players and $\psi^G$ encodes the guarantees required of the system. This is without loss of generality since, $\psi^A$ can always be taken to be true. As discussed in the introduction, solving problems in the form of (2) has applications in receding horizon control.

**Solution Technique.** There are numerous ways to algorithmically solve (2). We shall focus on a subset of the family of algorithms that fall under the CEGIS framework. This framework can be presented in a much more general setting, however, for the sake of exposition we present a formulation tailored to the controller-synthesis problem. We assume access to two oracles: a counterexample oracle $\mathcal{O}_{ce}$ which, given a candidate dominant strategy $\mathbf{u}$, returns a counter example $\mathbf{w}$ that refutes $\phi$; and a candidate oracle, $\mathcal{O}_{ca}$, which proposes a candidate dominant strategy given the previous counterexamples. CEGIS is then an iterative algorithm that repeatedly proposes new candidate strategies to $\mathcal{O}_{ce}$. If the candidate strategy is indeed dominant, the algorithm terminates and returns the strategy. Otherwise, $\mathcal{O}_{ce}$ returns a counterexample. If no candidate strategy is found, the algorithm terminates with failure to synthesize a winning strategy. The basic structure of the CEGIS meta algorithm is given in Fig 2.

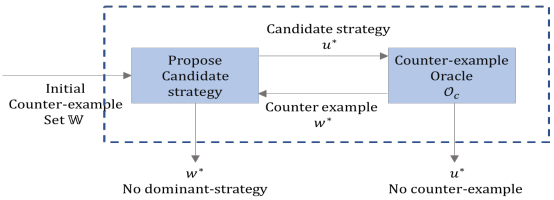


Fig. 2. CEGIS Meta algorithm.

Given this meta algorithm, one can choose $\mathcal{O}_{ca}$ and $\mathcal{O}_{ce}$ to define a new CEGIS variant. However, before designing $\mathcal{O}_{ca}$ and $\mathcal{O}_{ce}$, it is fruitful to characterize some requirements for soundness and completeness of a CEGIS scheme.

*Definition 1.* A counterexample oracle, $\mathcal{O}_{ce}$, is *sound* if: when given a candidate $\mathbf{u}$, if it returns $\perp$ then $\mathbf{u}$ is a dominant strategy, and if it returns $\mathbf{w}$ then $\mathbf{w}$ refutes $\mathbf{u}$, i.e., $\phi(\mathbf{u}, \mathbf{w})$ is False; $\mathcal{O}_{ce}$ is *complete* if: when given a $\mathbf{u}$ such that there exists a counterexample, it returns a counterexample.

*Definition 2.* A candidate oracle $\mathcal{O}_{ca}$ is *sound* if: when given the set of previous counterexamples $\mathbb{W}_k$, if $\mathcal{O}_{ca}$ returns $\perp$, then there is no strategy that simultaneously counters all moves in $\mathbb{W}_k$, and if it returns $\mathbf{u}$, then $\bigwedge_{\mathbf{w}\in\mathbb{W}_k} \phi(\mathbf{u}, \mathbf{w})$ is True; $\mathcal{O}_{ca}$ is *complete* if: given a counterexample set $\mathbb{W}_k$, if there exists $\mathbf{u}$ s.t. $\bigwedge_{\mathbf{w}\in\mathbb{W}_k} \phi(\mathbf{u}, \mathbf{w})$, then $\mathcal{O}_{ca}$ returns such a $\mathbf{u}$.

If $\mathbb{W}_k$ is equal to $\mathcal{W}$ then the conditions of Def 2 naturally define soundness and completeness of the CEGIS variant.

We notice that the soundness of the CEGIS algorithm follows immediately from the soundness of the oracles:

*Proposition 1.* (Soundness). Let $\mathbf{u}$ be the result of a CEGIS algorithm using sound oracles $\mathcal{O}_{ca}$ and $\mathcal{O}_{ce}$ for specification $\phi$. If $\mathbf{u} = \perp$ then there is no dominant strategy and if $\mathbf{u} \neq \perp$, then $\mathbf{u}$ is a dominant strategy.

In the sequel, we assume $\mathcal{O}_{ce}$ and $\mathcal{O}_{ca}$ are sound.

Similarly, we observe that if either $\mathcal{O}_{ce}$ or $\mathcal{O}_{ca}$ is not complete, then the CEGIS loop cannot possibly be complete (by completeness, we mean that if a dominant strategy exists, the algorithm will return it). Therefore, we also assume the oracles are complete.

This poses the question: *if $\mathcal{O}_{ce}$ and $\mathcal{O}_{ca}$ are sound and complete, does that imply that the CEGIS loop is complete?* The example below shows that the answer is negative.

*Example 4.* Consider the specification $\phi(\mathbf{u}, \mathbf{w}) = \mathbf{u} + \mathbf{w} \geq 0$ where $\mathbf{u}, \mathbf{w} \in [-1, 1]$. Clearly $\mathbf{u} = 1$ is a dominant strategy. Nevertheless, let us explore what could happen were we to solve this game algorithmically. Let $\mathcal{O}_{ce}$ and $\mathcal{O}_{ca}$ be sound and complete as defined above. Suppose we start the counterexample set with $\mathbb{W}_0 = \{0\}$. The candidate oracle may then propose $\mathbf{u} = 0$. The counterexample oracle can then refute with $\mathbf{w} = -0.1$, growing the counterexample set to $\mathbb{W}_1 = \{0, -0.1\}$. Then the system can propose $\mathbf{u} = 0.1$. In response the counter example oracle could refute with $-0.11$. Observe that this sequence of appending 1s can continue indefinitely, with no single example ever repeated. Thus, the CEGIS algorithm would never halt, despite having a dominant strategy.

Notice however, that if either $\mathcal{O}_{ce}$ or $\mathcal{O}_{ca}$ had returned values that minimize or maximize the quantitative semantics, $\rho(\mathbf{u}, \mathbf{w}) = \mathbf{u} + \mathbf{w}$, the above example would have terminated in a couple of iterations. This suggests forcing at least one of the oracles to return "optimal" (max or min $\rho$) candidates or counter examples, to ensure fewer iterations for termination. In this paper, we argue that $\mathcal{O}_{ce}$ should be optimal, but $\mathcal{O}_{ca}$ need not be. Intuitively, since the counterexample set $\mathbb{W}$ grows with each iteration, $\mathcal{O}_{ca}$ needs to solve a harder problem at each iteration, whereas the problem $\mathcal{O}_{ce}$ solves remains of the same difficulty.

*Is enforcing this optimality enough to guarantee completeness?* The answer is again negative.

*Example 5.* Consider a quantitative semantics such, that within a closed ball around $\mathbf{u}^* \in \mathcal{U}$, $\rho(\mathbf{u}, \mathbf{w}) = 0$. The same line of reasoning as in Ex 4 applies since no gradient exists within this ball and there are an infinite number of counterexamples. At this time, we do not know of any general conditions on $\rho$ and $\phi$ for completeness.

A consequence of making $\mathcal{O}_{ce}$ optimal is that one obtains an "anytime" algorithm by maintaining the input $\mathbf{u}$ with the best worst-case cost seen so far. This input is the closest to a dominant strategy that has been found. This is relevant for time bounded computations where optimal strategies, while preferable, may not be necessary.

Additionally, while we do not get completeness, we do get a weaker termination guarantee. *Suppose we are willing to accept a solution that leaves $\phi$ slightly unsatisfied. Namely, $\rho(\mathbf{u}, \cdot) \in [\delta, 0)$ for some $\delta < 0$. Will the algorithm terminate under this condition?* Theorem 2 answers this question in the affirmative.

*Theorem 2.* ($\delta$-Termination). If $\mathcal{U}$ is bounded, $\mathcal{O}_{ce}$ is optimal, and the quantitative semantics, $\rho(\mathbf{u}, \mathbf{w})$ satisfies (3) and (4), then either the CEGIS loop terminates or for any $\delta < 0$ there is an iteration $n \in \mathbb{N}$ such that $\forall \mathbf{w} \in \mathcal{W}\ .\ \rho(\mathbf{u}_n, \mathbf{w}) \geq \delta$.

*Lemma 1.* Let $\rho$ satisfy (3) and (4) with Lipschitz constant $L_\rho$. If $\rho(\mathbf{u}, \mathbf{w}) = \delta \neq 0$, then for all $\mathbf{u}'$ in the open ball of radius $|\delta/L|$ centered at $\mathbf{u}$, we have $\phi(\mathbf{u}, \mathbf{w}) = \phi(\mathbf{u}', \mathbf{w})$.

**Proof.** [Lem 1] Via Lipschitz Continuity (4), one must perturb $\mathbf{u}$ by at least $|\delta/L|$ to make $\rho(\mathbf{u}', \mathbf{w})$ zero. By (3) the sign of $\rho(\mathbf{u}', \mathbf{w})$ determines $\phi(\mathbf{u}', \mathbf{w})$. Thus, $\forall \mathbf{u}'$ in the open ball of radius $|\delta/L|$ centered at $\mathbf{u}$, $\phi(\mathbf{u}, \mathbf{w}) = \phi(\mathbf{u}', \mathbf{w})$.■

**Proof.** [Thm 2] Pick an arbitrary $\delta < 0$. Assume for contradiction that the CEGIS loop does not terminate, and for each iteration $i$, $\rho(\mathbf{u}_i, \mathbf{w}_i) < \delta$. Via Lem 1, the ball of radius $|\delta/L|$ of inputs around $\mathbf{u}_i$ is refuted by $\mathbf{w}_i$. Thus, at each iteration, including $\mathbf{w}_i$ in $\mathbb{W}_k$ refutes at least this ball around $\mathbf{u}_i$. Observe that $\mathcal{U}$ is bounded. Thus, there exists an iteration $k \in \mathbb{N}$ where $\mathcal{O}_{ca}$ must return $\perp$ since all of the input space has been refuted. This terminates the loop, leading to a contradiction. Therefore, either the CEGIS loops terminates or $\exists n \in \mathbb{N}$ such that $\rho(\mathbf{u}_n, \mathbf{w}_n) \geq \delta$. Further, since $\mathcal{O}_{ce}$ is optimal, $\rho(\mathbf{u}_n, \mathbf{w}_n) \geq \delta \implies \forall \mathbf{w} \in \mathcal{W},\ \rho(\mathbf{u}_n, \mathbf{w}) \geq \delta$.■

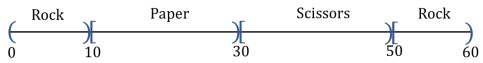


Fig. 3. Continuous RPS.

The proof of Thm 2 suggests a simple worst-case complexity of a CEGIS scheme.

*Theorem 3.* Let $(\mathcal{O}_{ce}, \mathcal{O}_{ca})$ be a CEGIS variant with an optimal $\mathcal{O}_{ce}$ and $\rho : [-1,1]^{n_u} \times [-1,1]^{n_w} \to \mathbb{R}$ satisfying (3) and (4) with Lipschitz constant $L_\rho$. If each CEGIS iteration takes at most $T$ steps, then the worst case running time for $\delta$-termination is:

$$O\left(T \cdot (L_\rho/\delta)^{n_u}\right) \tag{6}$$

**Proof.** The result follows directly from the fact that any two refuted balls can share at most half of their volume (since they cannot intersect each other's centers) and refuted balls have volume proportional to $(\delta/L_\rho)^{n_u}$.■

**Designing the Counterexample Oracle, $\mathcal{O}_{ce}$.** As previously stated, for each CEGIS variant, we need a counterexample oracle $\mathcal{O}_{ce}$ to solve the following problem:

$$\text{argmin}_{w \in \mathcal{W}} \{\rho(\mathbf{u}_k, \mathbf{w})\} \tag{7}$$

where $\mathbf{u}_k$ is the candidate strategy being refuted. In the experiments section, this will take the form of a MILP (Raman et al., 2014).

**Designing the Candidate Oracle.** In the CEGIS variants $\mathcal{C}_N$ and $\mathcal{C}_{SCE}$, $\mathcal{O}_{ca}$ was assumed to solve the following OP:

$$\text{argmax}_{\mathbf{u} \in \mathcal{U}} \left\{ \min_{\mathbf{w} \in \mathbb{W}_k} (\rho(\mathbf{u}, \mathbf{w})) \right\} \tag{8}$$

where $\mathbb{W}_k$ is a set of all previously seen counterexamples ($\mathcal{C}_N$) or the most recent counterexample ($\mathcal{C}_{SCE}$). In our experiments, (8) is a MILP (Raman et al., 2014).

In $\mathcal{C}_{SN}$, we assume the candidate oracle returns a $\mathbf{u}$ that solves the following satisfaction query.

$$\exists \mathbf{u} \in \mathcal{U} \,.\, \bigwedge_{\mathbf{w} \in \mathbb{W}_k} \phi(\mathbf{u}, \mathbf{w}) \tag{9}$$

In our experiments, this will be a satisfiability modulo theory (SMT) problem for Real Linear Arithmetic. [2] [3]

## 3. REFUTED INPUT SQUARES

Before developing the candidate oracle for $\mathcal{C}_{HYB}$, we give an example that concretely illustrates the problems with $\mathcal{C}_{SCE}$ and suggests a simple solution.

*Example 6.* Consider a continuous variant of the familiar zero-sum game: Rock, Paper, Scissors (RPS). Two players, the system ($s$) and the environment ($e$) simultaneously choose either ($R$)*ock*, ($P$)*aper* or ($S$)*cissors*. Let us represent the state of the system by $x$, the environment by $y$, the system move by $u$ and the environment move by $w$. The dynamics are given by $x = 60u$, $y = 60w$.

Fig 3 depicts the embedding of Rock, Paper, and Scissors onto $\mathbb{R}$. If the state of either player is in $(0, 10) \cup [50, 60)$, it is considered to be playing $R$; similarly, $[10, 30)$ is $P$ and $[30, 50)$ is $S$. Letting $A = (S, P, R)$, the system's winning condition is:

$$\bigwedge_{i=0}^{2} \Big( (y \in A[i \bmod 3)]) \implies (x \notin A[(i+1) \bmod 3]) \Big) \tag{10}$$

which encodes: *Rock* beats *Scissors*, *Scissors* beats *Paper*, and *Paper* beats *Rock*, respectively.

First observe that neither the system or the environment has a dominant strategy, and that for CEGIS to terminate the system must have a counterexample from each of the $R$, $P$ and $S$ regions in order to refute the entire input space. Thus, the loop will never terminate for $\mathcal{C}_{SCE}$.

Let $N$ denote the number of constraints required to encode a single counterexample in (8). The original motivation (Raman et al., 2014) for using $\mathcal{C}_{SCE}$ instead of $\mathcal{C}_N$ was that the size of (8) grew linearly with a rate at least as large as $N$. In this section, we explore a slight modification to (8) that grows linearly with a rate near the dimension of $\mathcal{U}$, which in general is much smaller than $N$.

One can think of the process of simultaneously handling the previously seen counterexamples as *implicitly* removing all parts of the input space that each counterexample refuted. We can also *explicitly* compute sets of refuted strategies ($\subset \mathcal{U}$) and modify the input space, $\mathcal{U}$ to exclude subsets containing the refuted candidates. Notice, however, that such sets are non-unique and potentially non-convex, so explicitly finding the maximal refuted set is not straightforward. Instead, inspired by Lemma 1, we choose to find the largest closed ball centered around $\mathbf{u}$ which is refuted by $\mathbf{w}$. More precisely, we seek to find the radius $r \in \mathbb{R}$ that solves:

$$\text{argmax}_{r \in \mathbb{R}} \{r \in \mathbb{R}_{>0} \mid \forall \mathbf{u} \in B_r(\mathbf{u}) \,.\, \neg\phi(\mathbf{u}, \mathbf{w})\} \tag{11}$$

where $B_r(\mathbf{u})$ is the closed ball of radius $r$ around $\mathbf{u}$. Denote the set of refuted balls by the $i$th round of CEGIS as $\mathcal{B}_i$. Then the candidate oracle for $\mathcal{C}_{HYB}$ solves:

$$\text{argmax}_{\mathbf{u} \in \mathcal{U}_k} \left\{ \min_{\mathbf{w} \in \mathbb{W}_k} (\rho(\mathbf{u}, \mathbf{w})) \right\} \tag{12}$$

where $\mathcal{U}_k = \mathcal{U} \setminus \bigcup \mathcal{B}_k$.

*Remark 3.* To approximately solve (11), first notice that given a candidate radius, one can use $\mathcal{O}_{ca}$ to query if a candidate exists within that radius. If there does, one must decrease the radius size. If not, one may increase the radius size. As the input space is bounded, one can then simply perform a binary search over the radii, and over approximate the optimal radius by an arbitrarily small margin. In our experiments, these series of queries are handled efficiently by an SMT engine.

Encoding a refuted ball needs roughly $2n_u$ constraints (where $\dim(\mathcal{U}) = n_u$). Further, if $\phi$ refers to each input at least once, $2n_u$ will be much smaller than $N$. Thus, this addresses the primary concern in developing $\mathcal{C}_{SCE}$. Given a counter example, we can now choose to allot either $N$ or $2n_u$ constraints. [4]

*Remark 4.* Due to the approximate nature of binary search on the real line, one must either err on the side of over- or under-approximation. We choose to err on the side over-approximation, potentially throwing out dominant strategies. A simple corollary of Lemma 1 is that if $\mathbf{w}$ is an optimal response to $\mathbf{u}$, $\rho(\mathbf{u}, \mathbf{w}) = a$, and $r$ is radius found by binary search, then all strategies, $u$, between $r$ and the true radius $r^*$ have $\rho(u, \mathbf{w}) \leq a + (r - r^*)/L$. Thus, if $(r - r^*)/L$ is at most $\epsilon$, one will only miss strategies that are not $\epsilon$ robust. Thus, when computing $r$, if one uses

[2] As discussed in the previous section, solving (9) rather than (8) comes at the cost of an optimal solution, something often desired in applications where a CEGIS loop is used. Nevertheless, we believe this solution can be made more robust with efficient post-processing.

[3] If the image of the semantics $\rho(\mathcal{U}, \mathcal{W})$ were $\{-1, 1\}$, then (9) and (8) are equivalent, but (4) does not hold.

[4] Let $i \in \mathbb{N}$ refer to the current iteration. In our implementation, we simultaneously keep the $k \in \mathbb{N}$ most recent counterexamples. The remaining $i - k$ candidate counterexample pairs are encoded as refuted balls. This trades-off encoding size for potentially more iterations.

an $\epsilon/(2L)$ tolerance and additionally adds $\epsilon/(2L)$ to the result, then one always over approximates $r^*$ by less than $\epsilon$. We refer to this approximation as "bloating by epsilon".

Observe that this provides a tunable completeness guarantee for $\mathcal{C}_{HYB}$:

*Theorem 4.* ($\epsilon$-Completeness). Given a bounded input space, $\mathcal{U}$, and $\rho$ satisfying (3) and (4), if there exists an $\epsilon$-robust solution, then $\mathcal{C}_{HYB}$ with $\epsilon$-bloating will find a dominant strategy.

**Proof.** At any iteration, $i$, of the CEGIS loop, let $\mathbf{u}_i$ and $\mathbf{w}_i$ be the candidate strategy and counterexample returned by $\mathcal{O}_{ca}$ and $\mathcal{O}_{ce}$, respectively. Due to $\epsilon$-bloating, the explicit refuted ball in each round has non-zero radius. As in the proof for Thm 2, since $\mathcal{U}$ is bounded, $\mathcal{C}_{HYB}$ must terminate. Further, by construction, $\epsilon$-bloating only omits strategies that are not $\epsilon$-robust. Therefore, if there exists an $\epsilon$-robust solution, then the final input space is non-empty. Thus, by exhaustive search, if there exists an $\epsilon$-robust solution, then $\mathcal{C}_{HYB}$ with $\epsilon$-bloating will find a dominant strategy. ■

## 4. EXPERIMENTS

In the following we benchmark the performance of the CEGIS variants $\mathcal{C}_{HYB}$, $\mathcal{C}_{SN}, \mathcal{C}_N$ and $\mathcal{C}_{SCE}$ across five experiments, each designed to vary the difficulty in a specific manner.

- In *Exp 1*, we extend the RPS game (Example 6) to study the effects of subdividing the input space into more regions (increased $N$) while keeping the number of counterexamples fixed.
- *Exp 2* extends Example 6 with additional moves. The number of counterexamples grows with the number of moves.
- In *Exp 3* we study the overhead of introducing refuted rectangles.
- In *Exp 4 and 5* we introduce linear dynamics to Example 6.

Our implementations of $\mathcal{C}_N$, $\mathcal{C}_{SCE}$, $\mathcal{C}_{HYB}$ and $\mathcal{C}_{SN}$ are available as a python toolbox MagnumSTL (Vazquez-Chanlatte, 2018). Our tool encodes the system and the specifications as a MILP using the encoding in (Raman et al., 2014). It uses two backend solvers, GLPK (Makhorin, 2018) for MILP (interfaced through optlang (Jensen et al., 2016)) and Z3 (De Moura and Bjørner, 2008) for SMT (interfaced through pysmt (Gario and Micheli, 2015)).

Further, as $\mathcal{C}_{SCE}$ cannot terminate in all but one experiment, for the purposes of comparison, we have omitted it. We consider two instances of $\mathcal{C}_{HYB}$ that remember the most recent and the two most recent counterexamples, resp.

**Experiment 1.** In this experiment we modify the embedding of $R$, $P$ and $S$ onto the real line (Fig 3) to that shown in Fig 4 where the domain is broken down into repeated $R - P - S$ segments.

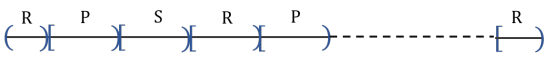


Fig. 4. Generalized continuous rps.

Notice that the required number of counterexamples remains three. However, for the $\mathcal{C}_{HYB}$, the number of iterations required increases as they must explicitly sample more $(\mathbf{u}, \mathbf{w})$ pairs. Our results are shown in Table 2.

We see that as predicted in (Raman et al., 2015), $\mathcal{C}_N$ scales poorly as the number of regions increase. This is due to the fact that the encoding size, $N$, increases as the number of regions increased. $\mathcal{C}_{HYB}$ with $k = 2$ scales only slightly

| # Regions | $\mathcal{C}_{SN}$ | $\mathcal{C}_N$ | $\mathcal{C}_{HYB}^{k=2}$ | $\mathcal{C}_{HYB}^{k=1}$ |
|---|---|---|---|---|
| 4 | 0.263 | 1.34 | 1.04 | 0.819 |
| 7 | 0.435 | 6.36 | 6.34 | 3.58 |
| 10 | 0.594 | 31 | 12.2 | 6.54 |

Table 2. Experiment 1 run times in seconds.

worse than $\mathcal{C}_{HYB}$ with $k = 1$ since, the MILP has to keep track of at least two counterexamples and is at least twice as big. $\mathcal{C}_{SN}$ performs the best, where the time taken remains more or less invariant to the number of regions.

**Experiment 2.** In this experiment, we generalize our RPS example to that shown in Fig 5: where the domain is broken

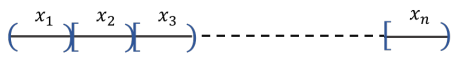


Fig. 5. Continuous RPS with $n$ counterexamples

into $n \geq 1$ possible plays, $x_1, \cdots, x_n$ such that $x_1$ beats $x_2$, $x_2$ beats $x_3$, ..., $x_n$ beats $x_1$.

In every round, the system wins if the following is satisfied:

$$\varphi_i = y[t] \in x_j \rightarrow x[t] \notin x_i \tag{13}$$

where $j = i + 1 \pmod n$.

Here, both the number of counterexamples as well as the number of $(\mathbf{u}, \mathbf{w})$ pairs generated by CEGIS iterations increase. Our results are shown in Table 3. We observe that the results follow the same trend as *Exp 1*.

| # Regions | $\mathcal{C}_{SN}$ | $\mathcal{C}_N$ | $\mathcal{C}_{HYB}^{k=2}$ | $\mathcal{C}_{HYB}^{k=1}$ |
|---|---|---|---|---|
| 1 | 0.039 | 0.061 | 0.062 | 0.061 |
| 2 | 0.155 | 0.394 | 0.392 | 0.360 |
| 3 | 0.350 | 1.68 | 1.14 | 0.676 |
| 4 | 0.604 | 6.01 | 2.26 | 1.34 |
| 5 | 0.951 | 180 | 3.81 | 1.98 |

Table 3. Experiment 2 run times in seconds.

**Experiment 3.** In this experiment, we consider a simple linear system, $x_{i+1} = x_i + \frac{5}{n}(u_i + w_i)$, and a high level specification, $\varphi = \bigvee_{i=0}^{n} x_i > 5$. Additionally, we have, $\mathcal{U} = \mathcal{W} = [-1, 1]$ and the initial state $x_0 = 0$. It is clear that the environment has a dominant strategy $w_i = -1$, and our conjecture is that we need only a few iterations for $\mathcal{C}_{HYB}$ to converge. Particularly, we study the overhead of introducing refuted rectangles. Our results are shown in Fig 6.

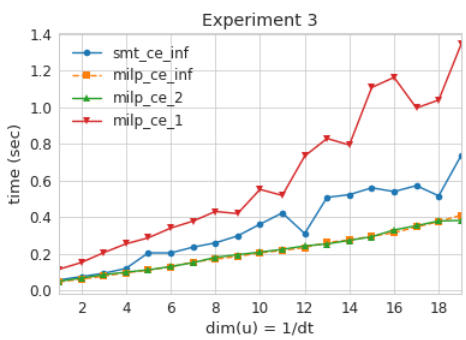


Fig. 6. The $x$-axis shows the dimension of the input space, while the $y$-axis shows the time taken for the CEGIS loop to converge. *milp_ce_1* refers to $\mathcal{C}_{HYB}$ with $k = 1$, *milp_ce_2* refers to $\mathcal{C}_{HYB}$ with $k = 2$, *milp_ce_inf* refers to $\mathcal{C}_N$, and *smt_ce_inf* refers to $\mathcal{C}_{SN}$.

All engines perform fairly well (compare with the times in *Exp 1 and 2*). As the original specification has a small encoding which is comparable to encoding the refuted rectangles, i.e., $N \approx 2n_u$, in this particular case, it appears better to encode the counterexample than the refuted input space.

**Experiments 4 and 5.** *Exp 4 and 5* are modifications to *Exp 1*. In *Exp 4*, we introduce linear dynamics to the RPS,

$$x_{i+1} = x_i + 30u_i/n \quad y_{i+1} = y_i + 30w_i/n \tag{14}$$

where $n$ is the number of steps taken and the initial state is ($x_0 = y_0 = 20$). The specifications on the system remain the same. As before, there is no dominant strategy for the system, and the number of counterexamples remains three. Our results are shown in Table 4. $\mathcal{C}_{HYB}$ with $k = 1$ performs reasonably well initially, but times out eventually (for discretization time $> 3$). $\mathcal{C}_{SN}$ outperforms all the other methods and remains more or less invariant to changes to discretization time. The timeouts for $\mathcal{C}_N$ and $\mathcal{C}_{HYB}$ with $k = 2$ happen after two counterexamples are found. The encoding then seems to become too large for GLPK to handle. For $k = 1$ the initial increase is due in part to the size of the MILP encoding, but also because more iterations are required since the counterexample in each iteration refutes less of the total input space.

| dim($u$) | $\mathcal{C}_{SN}$ | $\mathcal{C}_N$ | $\mathcal{C}_{HYB}^{k=2}$ | $\mathcal{C}_{HYB}^{k=1}$ |
|---|---|---|---|---|
| 1 | 0.35 | 18.01 | 1.52 | 0.88 |
| 2 | 0.53 | timeout | timeout | 5.11 |
| 3 | 1.00 | timeout | timeout | 6.38 |

Table 4. Experiment 4 run times in seconds.

In *Exp 5*, we introduce a small gap between $R$ and $P$ in *Exp 4*. We denote this to be the '*Spock*'. We update the specification $\phi$ with an additional specification,

$$\varphi_{dom} = (y[t] \in \{R, P, S\}) \rightarrow (x[t] \in Spock)$$

Choosing *Spock* yields a small positive $\rho$ against all disturbances and is, thus, a dominant strategy. Nevertheless, $\mathbf{u} \in Spock$ has $\rho$ so small that $\mathcal{O}_{ca}$ does not propose *Spock* unless $R$, $P$ and $S$ are refuted. $\mathcal{C}_N$ and $\mathcal{C}_{SN}$ would take at least three iterations to refute them.

| dim($u$) | $\mathcal{C}_{SN}$ | $\mathcal{C}_N$ | $\mathcal{C}_{HYB}^{k=2}$ | $\mathcal{C}_{HYB}^{k=1}$ |
|---|---|---|---|---|
| 1 | 0.206 | 1.23 | 1.25 | 0.773 |
| 2 | 0.321 | 3.57 | 3.61 | 7.28 |
| 3 | 0.662 | 28.31 | 33.40 | 39.00 |

Table 5. Experiment 5 run times in seconds.

Introducing the *Spock* region significantly improves performance. $\mathcal{C}_N$ and $\mathcal{C}_{HYB}$ with $k = 2$ no longer time out. We suspect that this is due to the dominant strategy aiding in the branch and bound heuristics the MILP solver uses. In practice, such a region will often exist. For example, this could correspond to a near miss angle in the example in Fig 1. Again, $\mathcal{C}_{SN}$ outperforms the others. Not shown are additional experiments increasing the time resolution by a factor of 12, at which point the optimality of $\mathcal{O}_{ce}$ leads to exponential blow up similar to the other variants.

## 5. CONCLUSION

In this paper, we take steps towards making CEGIS-based planning practical via a satisfaction oracle based on SMT, and incorporating refuted strategies along with counterexamples between iterations. We further present an empirical study suggesting that incorporating more than one counterexample while searching for new strategies is indeed scalable, and provide comparisons across different CEGIS implementations. CEGIS is an instance of oracle-guided inductive synthesis (Jha and Seshia, 2017), and our comparative study of CEGIS variants has similarities with that work. As future work, we would also like to (i) Leverage our new algorithms for real-time synthesis of robust receding horizon reactive strategies for complex cyber-physical systems with arbitrary temporal constraints. (ii) Leverage warm starts and incremental SMT engines to achieve further performance improvements. (iii) Further refine the convergence rate beyond termination guarantees. (iv) Leverage recent work on symbolic optimization with SMT solvers (Li et al., 2014).

## REFERENCES


De Moura, L. and Bjørner, N. (2008). Z3: An efficient SMT solver. *Tools and Algorithms for the Construction and Analysis of Systems.*

Ding, J., Li, E., Huang, H., and Tomlin, C.J. (2011). Reachability-based synthesis of feedback policies for motion planning under bounded disturbances. In *IEEE International Conference on Robotics and Automation.*

Farahani, S.S., Raman, V., and Murray, R.M. (2015). Robust model predictive control for signal temporal logic synthesis. *IFAC-PapersOnLine.*

Gario, M. and Micheli, A. (2015). PySMT: A solver-agnostic library for fast prototyping of SMT-based algorithms. In *13th International Workshop on Satisfiability Modulo Theories.*

Jensen, K., Cardoso, J., and Sonnenschein, N. (2016). Optlang: An algebraic modeling language for mathematical optimization. *Journal of Open Source Software.*

Jha, S. and Seshia, S.A. (2017). A Theory of Formal Synthesis via Inductive Learning. *Acta Informatica.*

Koymans, R. (1990). Specifying real-time properties with metric temporal logic. *Real-time systems.*

Li, Y., Albarghouthi, A., Kincaid, Z., Gurfinkel, A., and Chechik, M. (2014). Symbolic optimization with smt solvers. In *ACM SIGPLAN Notices.*

Makhorin, A.O. (2018). GLPK. `https://www.gnu.org/software/glpk/`.

Maler, O. and Nickovic, D. (2004). Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems.*

Mitchell, I.M., Bayen, A.M., and Tomlin, C.J. (2005). A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control.*

Raman, V., Donzé, A., Maasoumy, M., Murray, R.M., Sangiovanni-Vincentelli, A., and Seshia, S.A. (2014). Model predictive control with signal temporal logic specifications. In *IEEE 53rd Annual Conference on Decision and Control.*

Raman, V., Donzé, A., Sadigh, D., Murray, R.M., and Seshia, S.A. (2015). Reactive synthesis from signal temporal logic specifications. In *18th International Conference on Hybrid Systems: Computation and Control.*

Solar-Lezama, A., Tancau, L., Bodik, R., Seshia, S., and Saraswat, V. (2006). Combinatorial sketching for finite programs. *ACM SIGOPS Operating Systems Review.*

Vazquez-Chanlatte, M.J. (2018). MagnumSTL. `https://github.com/mvcisback/magnumSTL.git`.



Acknowledgments: We thank Markus Rabe and our anonymous reviewers for their invaluable feedback. This work is funded in part by the DARPA Assured Autonomy program, the DARPA BRASS program under agreement number FA8750-16-C-0043; the VeHICaL Project (NSF grant #1545126); the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-17-2-0196; Toyota under the iCyPhy center; Quantitative Contract-Based Synthesis and Verification for CPS Security (NSF grant #1739816).