
Proceedings Paper:
Dimitrova, R. and Finkbeiner, B. (2008) Abstraction refinement for games with incomplete information. In: Hariharan, R., Mukund, M. and Vinay, V., (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science. IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, 09-11 Dec 2008, Bangalore, India. Leibniz International Proceedings in Informatics (LIPIcs), 2 . Schloss Dagstuhl - Leibniz-Zentrum für Informatik , pp. 175-186. ISBN 9783939897088

https://doi.org/10.4230/LIPIcs.FSTTCS.2008.1751

Foundations of Software Technology and Theoretical Computer Science (Bangalore) 2008.
Editors: R. Hariharan, M. Mukund, V. Vinay; pp 175-186

Abstraction Refinement for Games with Incomplete Information*

Rayna Dimitrova†, Bernd Finkbeiner
Universität des Saarlandes
{dimitrova,finkbeiner}@cs.uni-sb.de

ABSTRACT.Counterexample-guided abstraction refinement (CEGAR) is used in automated software analysis to find suitable finite-state abstractions of infinite-state systems. In this paper, we extend CEGAR to games with incomplete information, as they commonly occur in controller synthesis and modular verification. The challenge is that, under incomplete information, one must carefully account for the knowledge available to the player: the strategy must not depend on information the player cannot see. We propose an abstraction mechanism for games under incomplete information that incorporates the approximation of the players' moves into a knowledge-based subset construction on the abstract state space. This abstraction results in a perfect-information game over a finite graph. The concretizability of abstract strategies can be encoded as the satisfiability of strategy-tree formulas. Based on this encoding, we present an interpolation-based approach for selecting new predicates and provide sufficient conditions for the termination of the resulting refinement loop.

1 Introduction

Infinite games are a natural model of reactive systems as they capture the ongoing interaction between a system and its environment. Many problems in automated software analysis, including controller synthesis and modular verification, can be reduced to finding (or deciding the existence of) a winning strategy. The design of algorithms for solving such games is complicated by the following two challenges: First, games derived from software systems usually have an infinite (or finite, but very large) state space. Second, the games are usually played under incomplete information: it is unrealistic to assume that a system has full access to the global state, e.g., that a process can observe the private variables of the other processes.

The most successful approach to treat infinite state spaces in software verification is predicate abstraction with counterexample-guided abstraction refinement (CEGAR) [3, 1]. For games with complete information [7, 4], CEGAR builds abstractions that overapproximate the environment's moves and underapproximate the system's moves. If the system wins the abstract game it is guaranteed to also win the concrete game. If the environment wins the abstract game, one checks if the strategy is spurious in the sense that it contains an abstract state from which the strategy cannot be concretized. If such a state exists, the state is split to ensure that the strategy is eliminated from further consideration.

For games with incomplete information, the situation is more complicated, because the strategic capabilities of a player depend not only on the available moves, but also on the

*This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center Automatic Verification and Analysis of Complex Systems (SFB/TR 14 AVACS).
†Supported by a Microsoft Research European PhD Scholarship and by an IMPRS-CS PhD Scholarship.

FSTTCS 2008
IARCS Annual Conference on
Foundations of Software Technology and Theoretical Computer Science
http://drops.dagstuhl.de/opus/volltexte/2008/1751

knowledge about the state of the game. If the abstract game provides less information to the system than the concrete game, then the environment may spuriously win the abstract game, because the abstract system may be unable to distinguish a certain pair of states and may therefore be forced to apply the *same* move in the two states where the concrete system can select *different* moves. An abstraction refinement approach for games with incomplete information must therefore carefully account for the information collected by the system. A first requirement is that the refinement should avoid predicates that mix variables that are observable to the system with those that are hidden. Such mixed predicates lead to the situation that the concrete system has partial information (the values of the observable variables), while the abstract system does not know the value of the predicate at all. Since the system may collect information over multiple steps of a play, however, just separating the variables alone is not enough. Consider, for example, a situation where, in order to win, the system has to react with output $x_o \approx 0$ if some hidden variable x_h has value $x_h \approx 0$ and with output $x_o \approx 1$ if $x_h \approx 1$. Now, suppose the system is able to deduce the value of x_h from the prefix that leads to the state, because an observable rational-valued variable x_i is either always positive or always negative if $x_h \approx 0$ and flips its sign otherwise. To rule out the spuriously winning strategy for the environment, it is necessary to refine the abstraction with the new predicate $x_i > 0$, even though the system wins for *any* value of x_i.

Contributions. In this paper, we propose the first CEGAR approach for games with incomplete information. We extend the abstraction of the game with a subset construction on the abstract state space that ensures that the system only uses information it can see. The result is a perfect-information game over a finite game graph that soundly abstracts the original game under incomplete information.

The refinement of the abstraction accounts for two cases: we *refine the abstract transition relations* by adding new predicates if the environment spuriously wins because it uses moves that are impossible in the concrete game or because moves of the system are impossible in the abstract game but possible in the concrete game; we *refine the observation equivalence* by adding new predicates if the environment spuriously wins because the abstract system has too little information. To ensure that the new predicates do not mix observable and unobservable variables, we develop a novel constraint-based interpolation technique which provides interpolants that meet arbitrary variable partitioning requirements.

The resulting refinement loop terminates for games for which a finite region algebra (that satisfies certain conditions related to the observation-equivalence) exists. This includes important infinite-state models such as timed games or games defined by bounded rectangular automata, given that the observation-equivalence meets the requirement.

In the following, due to space constraints, all proofs and some technical details have been omitted. We refer the reader to the full version of this paper [6].

Related work. The classic solution to games with incomplete information is the translation to perfect-information games with a *knowledge-based subset construction* due to Reif [9]. For games over infinite graphs, however, this construction is in general not effective. Our approach is symbolic and is therefore suited to the analysis of games over infinite state spaces. For incomplete-information games with finite state spaces, an alternative would be to first use the knowledge-based subset construction to obtain a perfect-information concrete game and then apply the CEGAR technique of [7] in the usual way. However, since the

subset construction leads to an exponential blow-up of the state space of the game, which for realistic systems will make the problem practically infeasible, it is imperative to first use predicate abstraction and obtain a much smaller state space and only then construct the subsets of observation-equivalent prefixes. *Symbolic fixed-point algorithms* based on antichains were proposed in [5, 2]. In the case of infinite game graphs, these algorithms are applied on a given finite region algebra for the infinite-state game. Our approach, on the other hand, automatically constructs a sufficiently precise finite abstraction. *Interpolation* was applied successfully in verification for the generation of refinement predicates. There one infers from an unconcretizable abstract counterexample-trace predicates, each of which refers only to variables that describe a single state on that trace. In our case we need to consider sets of traces each of which is concretizable and that are represented symbolically using sets of variables whose intersection contains observable variables only. The straightforward application of existing interpolation methods ([8, 10]) would produce refinement predicates that are either guaranteed to be observable or guaranteed not to relate two or more states. These approaches are incapable of meeting both guarantee requirements. To this end, we present our extension of the algorithm from [10] which provides interpolants that meet arbitrary variable partitioning requirements.

2 Preliminaries

Variables, predicates and formulas. We model the communication between a system and its environment with a finite set X of *variables*, which is partitioned into four pairwise disjoint sets: X_h, X_i, X_o and $\{t\}$. The environment updates (and can observe) the variables in X_h and X_i and the system updates (and can observe) the variables in X_o. The variables in X_i are the input variables for the system, i.e., it can read their value but not update them. The variables in X_h are private variables for the environment, i.e., the system cannot even observe them. The set X_o consists of the output variables of the system which can be only read by the environment. The value of the auxiliary variable t determines whether it is the system's or the environment's turn to make a transition, i.e., the two players take turns in making a transition. The set X' consists of the primed versions of the variables in X.

Sets of concrete and abstract states and transitions are represented as *formulas* over some possibly infinite set $\mathcal{AP}$ of *predicates* (*atomic formulas*) over the variables in $X \cup X'$. For a formula φ, we denote with $Vars(\varphi)$ and $Preds(\varphi)$ the sets of variables and predicates, respectively, that occur in φ. For a set $\mathcal{P}$ of predicates, the set $Obs(\mathcal{P})$ consists of the predicates in $\mathcal{P}$ that contain only observable variables, i.e., from $Obs(X \cup X') = (X \cup X') \setminus (X_h \cup X'_h)$.

Game structures. A *game structure with perfect information* $C = (S_s, S_e, s_0, R_s, R_e)$ consists of a set of states $S = S_s \cup S_e$, which is partitioned into a set S_s of *system states* and a set S_e of *environment states*, a distinguished initial state $s_0 \in S$, and a transition relation $R = R_s \cup R_e$, where $R_s \subseteq S_s \times S_e$ (when the system makes a transition, it always gives back the turn to the environment) and $R_e \subseteq S_e \times S$ are the transition relations for the system and the environment respectively. A *game structure with incomplete information* $(S_s, S_e, s_0, \equiv, R_s, R_e)$ additionally defines an *observation equivalence* $\equiv$ on S. The system has partial knowledge about the current state, i.e., it knows the equivalence class of the current state, but not the particular state in this class. We require that the relation $\equiv$ meets the following two require-

ments. The relation $\equiv$ respects the partitioning of S into S_s and S_e: If $v_1 \in S_s$ and $v_2 \in S_e$ then $v_1 \not\equiv v_2$. The system can distinguish between the different successors of a system state: For every $v \in S_s$ and $w_1, w_2 \in S_e$, if $(v, w_1) \in R_s$, $(v, w_2) \in R_s$ and $w_1 \neq w_2$, then $w_1 \not\equiv w_2$. The set of available transitions in a system state is the same for all observation-equivalent states: For every states $v_1, v_2 \in S_s$ and $w_1 \in S_e$ such that $v_1 \equiv v_2$ and $(v_1, w_1) \in R_s$, there exists a state $w_2 \in S_e$ such that $w_1 \equiv w_2$ and $(v_2, w_2) \in R_s$. A state v for which there is no $w \in S$ with $(v, w) \in R$ is called a *dead-end*.

We use a symbolic representation of game structures. A *symbolic game structure with incomplete information* $\mathcal{C} = (X, init, \mathcal{T}_s, \mathcal{T}_e)$ consists of a set of variables X (partitioned into X_h, X_i, X_o and $\{t\}$), a formula *init* over X and formulas $\mathcal{T}_s$ and $\mathcal{T}_e$ over $X \cup X'$. For simplicity, we assume that we have singleton sets $X_h = \{x_h\}$, $X_i = \{x_i\}$ and $X_o = \{x_o\}$ (the extension to the general case is trivial). The formulas are required to satisfy the following conditions: (1) $\mathcal{T}_e$ implies $t \approx 0$ and $x'_o \approx x_o$, (2) $\mathcal{T}_s$ implies $t \approx 1$, $t' \approx 0$, $x'_h \approx x_h$ and $x'_i \approx x_i$, (3) the formula $\mathcal{T}_s\{x_h \mapsto x_h^1\} \leftrightarrow \mathcal{T}_s\{x_h \mapsto x_h^2\}$ is valid.

Let H, I and O be the domains of x_h, x_i and x_o respectively. We assume that the set O of possible *outputs* for the system is finite. We denote with c_o the constant from the signature corresponding to an element $o \in O$ and with C_o the set of all constants for elements of O. The domain of t is $\{0, 1\}$. The set $Val(X)$ consists of all total functions that map each variable in X to its domain. For a formula φ over X, and $v \in Val(X)$ we denote with $\varphi[v]$ the truth value of the formula φ for the valuation v of the variables. We write $v \models \varphi$ iff $\varphi[v]$ is true. For a formula φ over $X \cup X'$, $v \in Val(X)$ and $w \in Val(X')$, $\varphi[v, w]$ is defined analogously.

A symbolic game structure $\mathcal{C} = (X, init, \mathcal{T}_s, \mathcal{T}_e)$ together with corresponding variable domains defines a game structure with incomplete information $C = (S_s, S_e, s_0, \equiv, R_s, R_e)$ in the following way. The sets S_s and S_e consist of the valuations in $Val(X)$ where t is mapped to 1 and 0 respectively. Since the variable x_h cannot be observed by the system, two states are observation-equivalent if they agree on the valuation of the variables in $Obs(X)$. We require that *init* is satisfied by a single initial state s_0. The formulas $\mathcal{T}_s$ and $\mathcal{T}_e$ define the transition relations, where $(v, w) \in R_s$ iff $\mathcal{T}_s[v, w]$ is true, and R_e is defined analogously.

For a formula φ and $c_o \in C_o$, $Pre_s(c_o, \varphi)$ is a formula such that $v \models Pre_s(c_o, \varphi)$ iff there exists $w \models \varphi \wedge x_o \approx c_o$ such that $(v, w) \in R_s$, $Pre_s(\varphi) = \bigvee_{c_o \in C_o} Pre_s(c_o, \varphi)$ and $Pre_e(\varphi)$ is a formula such that $v \models Pre_e(\varphi)$ iff there exists $w \models \varphi$ such that $(v, w) \in R_e$.

Safety games. We consider safety games defined by a set of *error states*, which we assume w.l.o.g. to be a subset of S_e. The objective for the system is to avoid the error states. Clearly, w.l.o.g. we can assume that S does not contain dead-ends and that for every $v \in S_s$ and $c_o \in C_o$, $v \models Pre_s(c_o, true)$. A *safety game* with perfect information (with incomplete information) $G = (C, E)$ consists of a game structure C with complete information (with incomplete information) and a set of error states E. A *symbolic safety game* $\mathcal{G} = (\mathcal{C}, err)$ consists of a symbolic game structure $\mathcal{C}$ and a formula *err* denoting the set of error states.

Strategies. Let G be a safety game. A *path* in G is a finite sequence $\pi = v_0 v_1 \dots v_n$ of states such that for all $0 \leq j < n$, we have $(v_j, v_{j+1}) \in R$. The length $|\pi|$ of π is $n + 1$. For $0 \leq j < |\pi|$, $\pi[j]$ is the j-th element of π and $\pi[0, j] = v_0 \dots v_j$. We define $last(\pi) = \pi[|\pi| - 1]$. A *prefix* in G is a path $\pi = v_0 v_1 \dots v_n$ such that $v_0 = s_0$. We call π a *system prefix* if $last(\pi) \in S_s$, and an *environment prefix* otherwise. We denote with $Prefs(G)$ the set of prefixes in G, and with $Prefs_s(G)$ and $Prefs_e(G)$ the sets of system and environment prefixes,

respectively. A *play* in G is either an infinite sequence $\omega = v_0 v_1 \dots v_j \dots$ with $v_0 = s_0$ and for all $j \geq 0$, $(v_j, v_{j+1}) \in R$ or a prefix π such that $last(\pi)$ is an error state. For an infinite play ω, $|\omega| = \infty$. The observation-equivalence $\equiv$ can be extended in a natural way to prefixes and plays. A *strategy for the system* is a function $f_s : Prefs_s(G) \to S_e$ such that if $f_s(\pi) = v$, then $(last(\pi), v) \in R_s$. Strategies for the environment are defined analogously. A strategy f_s for the system in an incomplete-information game is called *consistent* iff for all $\pi_1, \pi_2 \in Prefs_s(G)$ with $\pi_1 \equiv \pi_2$, it holds that $f_s(\pi_1) \equiv f_s(\pi_2)$. The outcome of two strategies f_s and f_e is a play $\omega = Outcome(f_s, f_e)$ such that for all $0 \leq j < |\omega|$ if $\omega[j] \in S_s$ then $\omega[j+1] = f_s(\omega[0, j])$ and if $\omega[j] \in S_e$ then $\omega[j+1] = f_e(\omega[0, j])$. A strategy f_s for the system is *winning* iff for every strategy f_e for the environment, if $\omega = Outcome(f_s, f_e)$ then for every $j \geq 0$, $\omega[j]$ is not an error state. A strategy f_e for the environment is winning iff for every strategy f_s for the system, if $\omega = Outcome(f_s, f_e)$ then for some j, $\omega[j]$ is an error state.

Strategy trees. A winning strategy f_e for the environment in a safety game G can be naturally represented as a finite tree $T(f_e)$, called *strategy tree*. Each node in $T(f_e)$ is labeled by a state in S, such that the following are satisfied: (1) the root is labeled by the initial state s_0, (2) if an internal node is labeled by a state v and a child of that node is labeled by a state w, then $(v, w) \in R$, (3) if an internal node π is labeled by $v \in S_s$, then for every $w \in S$ with $(v, w) \in R_s$, there exists exactly one child of π which is labeled by w, and $Children(\pi, T(f_e))$ is the set of all children of π in $T(f_e)$, (4) if an internal node π is labeled by $v \in S_e$, then that node has exactly one child, denoted by $Child(\pi, T(f_e))$, labeled by some $w \in S$ with $(v, w) \in R_e$, (5) a node is a leaf iff it is labeled by an error state. Thus, each node corresponds to a prefix in G, and a prefix in $Prefs(G)$ is represented by at most one node. We identify each node with the corresponding prefix and define $Prefs(f_e)$ as the set of prefixes in $T(f_e)$.

Knowledge-based subset construction. The *knowledge-based subset construction* of an incomplete-information game $G = ((S_s, S_e, s_0, \equiv, R_s, R_e), E)$ is a perfect-information game $G^k = ((S_s^k, S_e^k, s_0^k, R_s^k, R_e^k), E^k)$ defined as follows: $S_s^k = \{V \in 2^{S_s} \setminus \{\emptyset\} \mid \forall v_1, v_2 \in V.\ v_1 \equiv v_2\}$; $S_e^k = \{V \in 2^{S_e} \setminus \{\emptyset\} \mid \forall v_1, v_2 \in V.\ v_1 \equiv v_2\}$; $s_0^k = \{s_0\}$; $(V, W) \in R_s^k$ iff $V \in S_s^k$, $W \in S_e^k$ and (1) for every $v \in V$ there is a $w \in W$ such that $(v, w) \in R_s$, (2) for every $w \in W$ there is a $v \in V$ such that $(v, w) \in R$ and (3) if $w_1 \equiv w_2$, $w_1 \in W$ and there is a $v \in V$ with $(v, w_2) \in R$ then $w_2 \in W$; $(V, W) \in R_e^k$ iff $V \in S_e^k$, $W \in S_s^k \cup S_e^k$ and (1′), (2) and (3) are satisfied, where (1′) there exist $v \in V$ and $w \in W$ such that $(v, w) \in R_e$; $E^k = \{V \in S_e^k \mid V \cap E \neq \emptyset\}$.

The game solving problem. The *game solving problem* is to determine whether there exists a consistent winning strategy for the system player in a given safety game with incomplete information. The *strategy synthesis problem* is to find such a strategy if one exists.

3 Abstraction

We use two subset constructions to abstract infinite-state games with incomplete information into finite-state games with perfect information: first, we overapproximate the moves of the environment and underapproximate the moves of the system in the abstract domain defined by the predicate valuations. Then, we overapproximate the observation-equivalence based on the observable predicates to obtain a sound abstraction.

Let $\mathcal{G} = (\mathcal{S}, err)$ be a symbolic safety game. For a finite set of predicates $\mathcal{P}$ over X, $Vals(\mathcal{P})$ is the set of all valuations of the elements of $\mathcal{P}$. For $a \in Vals(\mathcal{P})$, and $p \in \mathcal{P}$, $[a]$ is

the corresponding formula over $\mathcal{P}$ and we write $a \models p$ iff the value of p in a is *true*. Similarly for a formula φ over $\mathcal{P}$. The concretization $\gamma_{\mathcal{P}}(a)$ of $a \in \mathit{Vals}(\mathcal{P})$ is the set of concrete states $\{s \in S \mid \forall p \in \mathcal{P} : s \models p \text{ iff } a \models p\}$. For $A \subseteq \mathit{Vals}(\mathcal{P})$, we define $\gamma_{\mathcal{P}}(A) = \bigcup_{a \in A} \gamma_{\mathcal{P}}(a)$. For a_1 and a_2 in $\mathit{Vals}(\mathcal{P})$, we define $a_1 \equiv^a_{\mathcal{P}} a_2$ iff for every $p \in \mathit{Obs}(\mathcal{P})$, $a_1 \models p$ iff $a_2 \models p$.

We abstract a concrete game w.r.t. a pair $\mathcal{P} = (\mathcal{P}_{se}, \mathcal{P}_s)$ of finite sets of predicates such that $\mathit{Preds}(\mathit{init}) \cup \mathit{Preds}(\mathit{err}) \cup \{t \approx 0\} \subseteq \mathcal{P}_{se}$. The states in S_e are abstracted w.r.t. $\mathcal{P}_{se}$ and the states in S_s are abstracted w.r.t. the full set $\mathcal{P} = \mathcal{P}_{se} \cup \mathcal{P}_s$. We require that $\mathit{PredsSyst}(\mathcal{P}_{se}) \subseteq \mathcal{P}_s$, where $\mathit{PredsSyst}(\mathcal{Q}) = \bigcup_{a \in \mathit{Vals}(\mathit{Obs}(\mathcal{Q}))} \mathit{Preds}(\mathit{Pre}_s([a]))$, to ensure the absence of dead-ends in the abstract game. By refining $\mathcal{P}_s$ with predicates that are used to split only abstract system states, we ensure the monotonicity of the abstraction of R_s. In the following, $\gamma(a)$ means $\gamma_{\mathcal{P}_{se}}(a)$ if $a \in \mathit{Vals}(\mathcal{P}_{se})$ and $\gamma_{\mathcal{P}}(a)$ if $a \in \mathit{Vals}(\mathcal{P})$. Similarly for $\equiv^a$.

For two pairs of sets of predicates $\mathcal{P} = (\mathcal{P}_{se}, \mathcal{P}_s)$ and $\mathcal{Q} = (\mathcal{Q}_{se}, \mathcal{Q}_s)$, we write $\mathcal{P} \subseteq \mathcal{Q}$ iff $\mathcal{P}_{se} \subseteq \mathcal{Q}_{se}$ and $\mathcal{P}_s \subseteq \mathcal{Q}_s$, and define $\mathcal{P} \cup \mathcal{Q} = (\mathcal{P}_{se} \cup \mathcal{Q}_{se}, \mathcal{P}_s \cup \mathcal{Q}_s)$.

The *abstraction* $\alpha(\mathcal{G}, \mathcal{P})$ of $\mathcal{G} = (\mathcal{S}, \mathit{err})$ w.r.t. a pair $\mathcal{P} = (\mathcal{P}_{se}, \mathcal{P}_s)$ of finite sets of predicates is the perfect-information safety game $G^a = ((S^a_s, S^a_e, s^a_0, R^a_s, R^a_e), E^a)$ defined below.

States. The set S^a of abstract states is the union of $S^a_s \subseteq 2^{\mathit{Vals}(\mathcal{P})} \setminus \{\emptyset\}$ and $S^a_e \subseteq 2^{\mathit{Vals}(\mathcal{P}_{se})} \setminus \{\emptyset\}$ which are defined as follows. An element A of $2^{\mathit{Vals}(\mathcal{P})} \setminus \{\emptyset\}$ belongs to S^a_s iff (1) for every $a \in A$, $a \not\models t \approx 0$ and $\gamma(a) \neq \emptyset$ and (2) for every $a_1, a_2 \in A$, $a_1 \equiv^a a_2$. Similarly, an element A of $2^{\mathit{Vals}(\mathcal{P}_{se})} \setminus \{\emptyset\}$ belongs to S^a_e iff (1) for every $a \in A$, $a \models t \approx 0$ and $\gamma(a) \neq \emptyset$ and (2) for every $a_1, a_2 \in A$, $a_1 \equiv^a a_2$. The initial abstract state s^a_0 consists of the single element a_0 of S^a such that $a_0 \models \mathit{init}$ and $\gamma(a_0) \neq \emptyset$.

May transitions. The abstract transition relation $R^a_e \subseteq S^a_e \times S^a$ for the environment is defined as: $(A, A') \in R^a_e$ iff the following are satisfied: (1 **may**) there exist $a \in A$, $v \in \gamma(a)$, $a' \in A'$ and $v' \in \gamma(a')$ with $(v, v') \in R_e$, (2) for every $a' \in A'$ there exist $a \in A$, $v \in \gamma(a)$ and $v' \in \gamma(a')$ such that $(v, v') \in R$ and (3) for every $a'_1 \in \mathit{Vals}(\mathcal{P})$ and $a'_2 \in \mathit{Vals}(\mathcal{P})$, if $a'_1 \in A'$, $a'_1 \equiv^a a'_2$ and there exist $a \in A$, $v \in \gamma(a)$ and $v' \in \gamma(a'_2)$ with $(v, v') \in R$, then $a'_2 \in A'$.

Must transitions. The abstract transition relation $R^a_s \subseteq S^a_s \times S^a_e$ for the system is defined as: $(A, A') \in R^a_s$ iff the conditions (1 **must**), (2) and (3) are satisfied, where: (1 **must**) for every $a \in A$ and every $v \in \gamma(a)$ there exist $a' \in A'$ and $v' \in \gamma(a')$ with $(v, v') \in R_s$.

Error states. An abstract state A is an element of E^a iff there exists an $a \in A$ with $a \models \mathit{err}$.

Concretization. The concretization $\gamma^k(f_e)$ of a winning strategy f_e for the environment in G^a is a set of winning environment strategies in the knowledge-based game G^k. For $\pi \in \mathit{Prefs}(G^a)$, we define $\gamma^k(\pi) = \{\pi^k \in \mathit{Prefs}(G^k) \mid |\pi^k| = |\pi|, \forall j : 0 \leq j < |\pi| \Rightarrow \pi^k[j] \subseteq \gamma(\pi[j])\}$ and $\gamma(\pi) = \{\pi^c \in \mathit{Prefs}(G) \mid |\pi^c| = |\pi|, \forall j : 0 \leq j < |\pi| \Rightarrow \pi^c[j] \in \gamma(\pi[j])\}$ (similarly for paths). Then $\gamma^k(f_e)$ is the set of all winning environment strategies f^k_e in G^k such that for every $\pi^k \in \mathit{Prefs}(f^k_e)$ there exists $\pi \in \mathit{Prefs}(f_e)$ with $\pi^k \in \gamma^k(\pi)$. Let $\mathcal{P}$ and $\mathcal{Q}$ be pairs of sets of predicates with $\mathcal{P} \subseteq \mathcal{Q}$. If π and π' are prefixes in $\alpha(\mathcal{G}, \mathcal{P})$ and $\alpha(\mathcal{G}, \mathcal{Q})$, respectively, we write $\pi' \leq \pi$ iff $|\pi| = |\pi'|$ and for every $0 \leq j < |\pi|$, $\gamma(\pi'[j]) \subseteq \gamma(\pi[j])$. If f_e and f'_e are winning strategies for the environment in $\alpha(\mathcal{G}, \mathcal{P})$ and $\alpha(\mathcal{G}, \mathcal{Q})$ respectively, then $f'_e \leq f_e$ iff for every $\pi' \in T(f'_e)$ there exists $\pi \in T(f_e)$ such that $\pi' \leq \pi$.

THEOREM 1. *[Soundness of the abstraction] If f_s is a winning strategy for the system in the perfect-information game $\alpha(\mathcal{G}, \mathcal{P})$, then there exists a consistent winning strategy f^c_s for the system in the symbolic game $\mathcal{G}$ with incomplete information.*

4 Abstract Counterexample Analysis

A winning strategy f_e for the environment in the game $\alpha(\mathcal{G}, \mathcal{P})$ is a *genuine* counterexample if it has a winning concretization in G^k. Otherwise it is called *spurious*. The analysis of the strategy-tree $T(f_e)$ constructs a *strategy-tree formula* $F(f_e)$ that is satisfiable iff f_e is genuine. The key idea is to symbolically simulate a perfect-information game over the equivalence classes of the prefixes of the concrete game structure G with incomplete information.

Traces and error paths. With each node π in $T(f_e)$, we associate a set $\mathit{Traces}(\pi)$ of traces, where a *trace* is a finite sequence $\tau \in C_o^*$ of system outputs, and define $\mathit{Traces}(f_e) = \mathit{Traces}(s_0^a)$. Each trace induces a set of concrete error paths in G. If the strategy f_e is genuine, then for each $\tau \in \mathit{Traces}(f_e)$, the concrete strategy in G^k should provide an error path ξ_τ in G. If π is a leaf node (i.e., an error node), then $\mathit{Traces}(\pi) = \{\epsilon\}$, otherwise, if π is a system node, then $\mathit{Traces}(\pi) = \{c_o\tau \mid c_o \in C_o, \rho \in \mathit{Children}(\pi, T(f_e)), \tau \in \mathit{Traces}(\rho)\}$, and, if π is an environment node, then $\mathit{Traces}(\pi) = \mathit{Traces}(\mathit{Child}(\pi, T(f_e)))$. A path ξ in the concrete game structure G is an *error path of a trace* τ if one of the following three conditions is satisfied: (i) $\xi[0] \models \mathit{err}$, (ii) $\xi[0] \in S_e$ and $\xi[1, |\xi|]$ is an error path for τ or (iii) $\xi[0] \in S_s$, $\tau = c_o\sigma$, $\xi[1]$ is a c_o-successor of $\xi[0]$ and $\xi[1, |\xi|]$ is an error path for σ.

Trace formulas. For each $\tau \in \mathit{Traces}(f_e)$ we define a formula $F(f_e, \tau)$ which is satisfiable iff there is a node $\rho \in T(f_e)$ such that there is an error path for τ in $\gamma(\rho)$. Here, unlike in the perfect-information case, in the concrete strategy the error paths ξ_{τ_1} and ξ_{τ_2} for two different traces $\tau_1, \tau_2 \in \mathit{Traces}(f_e)$ may differ even before the first position in which τ_1 and τ_2 are different, as long as their prefixes up to that position are equivalent. We encode this constraint by indexing the variables in the trace formulas as explained below.

Consider a node π and a trace $\tau \in \mathit{Traces}(f_e)$ such that $\tau = \sigma_1\sigma_2$, σ_1 corresponds to the outputs on the prefix π and $\sigma_2 \in \mathit{Traces}(\pi)$. The variables in $F(f_e, \tau)$ are indexed as follows. The variables that represent a concrete state in $\gamma(\mathit{last}(\pi))$ are indexed with the node π, so that there are different variables in the formula for different nodes. They are indexed also with the part σ_1 of τ, so that there are different variables in different trace formulas after the first difference in the outputs. The unobservable variables have to be indexed additionally with the remaining part σ_2 of τ, in order to have different unobservable variables for corresponding states in different trace formulas even before the first difference in the outputs. To this end, with each node $\pi \in T(f_e)$ and $\sigma_1, \sigma_2 \in C_o^*$ we associate a set $X^{(\pi,\sigma_1,\sigma_2)} = \{x_h^{(\pi,\sigma_1,\sigma_2)}, x_i^{(\pi,\sigma_1)}, x_o^{(\pi,\sigma_1)}, t^{(\pi,\sigma_1)}\}$ of variables and define substitutions which map variables from the original set $X \cup X'$ to variables in the sets $X^{(\pi,\sigma_1,\sigma_2)}$ and vice versa.

We define recursively a *trace formula* $F(\pi, \tau)$ for every node $\pi \in T(f_e)$ and trace $\tau \in \mathit{Traces}(\pi)$. We consider three cases that correspond to the three cases in the definition of error paths: the auxiliary formulas *ErrorState*, *EnvTrans* and *SystTrans* account for cases (i), (ii) and (iii) respectively. If π is a leaf node, then $\tau = \epsilon$ and $F(\pi, \epsilon) = \mathit{ErrorState}(\pi)$. If π is an internal environment node we define $F(\pi, \tau) = \mathit{EnvTrans}(\pi, \pi', \tau)$, where $\pi' = \mathit{Child}(\pi, T(f_e))$. If π is an internal system node, then $\tau = c_o\sigma$ for some c_o and σ and we define $F(\pi, c_o\sigma) = \bigvee_{\pi' \in \mathit{Children}(\pi, T(f_e))} \mathit{SystTrans}(\pi, \pi', c_o\sigma)$. By the definition, the trace formula $F(\pi, \tau)$ is satisfied by a sequence ξ of concrete states iff ξ is an error path for τ and there exists a node ρ in the subtree of $T(f_e)$ below π such that $\xi \in \gamma(\rho)$.

Strategy-tree formula. We define $F(f_e, \tau) = F(s_0^a, \tau)$ for every $\tau \in \mathit{Traces}(f_e)$ and finally, the

strategy-tree formula is $F(f_e) = \bigwedge_{\tau \in Traces(f_e)} F(f_e, \tau)$. It can be constructed by annotating in a bottom-up manner the nodes in $T(f_e)$ with the corresponding sets of traces and formulas.

THEOREM 2. *Let f_e be a winning strategy for the environment in the game $\alpha(\mathcal{G}, \mathcal{P})$. The formula $F(f_e)$ is satisfiable iff the strategy f_e is genuine, i.e., iff $\gamma^k(f_e) \neq \emptyset$.*

5 Counterexample-Guided Refinement

If f_e is a spurious winning strategy for the environment in $\alpha(\mathcal{G}, \mathcal{P})$, we enhance $\mathcal{P} = (\mathcal{P}_{se}, \mathcal{P}_s)$ with sets of refinement predicates $R_{se}(f_e)$ and $R_s(f_e)$, such that in the refined game $\alpha(\mathcal{G}, (\mathcal{P}_{se} \cup R_{se}(f_e), \mathcal{P}_s \cup R_s(f_e)))$ the environment has no winning strategy f'_e with $f'_e \leq f_e$.

5.1 Refining the Abstract Transition Relations

If for some $\tau \in Traces(f_e)$ the formula $F(f_e, \tau)$ is unsatisfiable, then the occurrence of the spurious abstract strategy is due to the approximations of the transition relations. Therefore we compute refinement predicates for eliminating the approximations that cause the existence of f_e. Such predicates can be determined by a bottom-up analysis of the strategy tree $T(f_e)$ that annotates each node π in the tree with a formula $\tilde{F}(\pi, \tau)$ for each trace $\tau \in Traces(\pi)$. The formula $\tilde{F}(\pi, \tau)$ denotes the subset of $\gamma(\pi)$ that consist of those concrete states from which there exists a concrete path that satisfies $F(f_e, \tau)$. We denote with $\texttt{RPGG}(f_e)$ (Refinement Predicates for the Game Graph) the pair $(\texttt{RPGG}_{se}(f_e), \texttt{RPGG}_s(f_e))$ of sets of predicates computed at this step and used to enhance $\mathcal{P}_{se}$ and $\mathcal{P}_s$, respectively.

State formulas. For $\pi \in T(f_e)$ and $\tau \in Traces(\pi)$, we define $\tilde{F}(\pi, \tau)$ as follows. If π is a leaf node, then $\tau = \epsilon$ and $\tilde{F}(\pi, \epsilon) = \bigvee_{a \in last(\pi), a \models err} [a]$. Otherwise, $\tau = c_o \sigma$ and $\tilde{F}(\pi, \tau) = [last(\pi)] \wedge Pre_s(c_o, \bigvee_{\pi' \in Children(\pi, T(f_e))} \tilde{F}(\pi', \sigma))$ if π is a system node, and $\tilde{F}(\pi, \tau) = [last(\pi)] \wedge Pre_e(\tilde{F}(Child(\pi, T(f_e)), \tau))$ otherwise. If $\tau \notin Traces(\pi)$, then $\tilde{F}(\pi, \tau)$ is $\tilde{F}(\pi, \sigma)$, where σ is the maximal prefix of τ such that $\sigma \in Traces(\pi)$ if such exists, and *false* otherwise.

Refinement predicates. The set $\texttt{RPGG}_{se}(f_e)$ of predicates with which we enhance $\mathcal{P}_{se}$ contains all predicates that occur in the annotation formulas $\tilde{F}(\pi, \tau)$. We ensure that the refined abstraction is precise w.r.t. the outputs from some trace $\tau \in Traces(f_e)$ for which $\tilde{F}(f_e, \tau)$ is unsatisfiable, by adding the elements of $OutPreds(\{\tau\})$ for one such τ to $\texttt{RPGG}_{se}(f_e)$, where for a set T of traces, we have defined $OutPreds(T) = \{x_o \approx \tau[j] \mid \tau \in T, 0 \leq j < |\tau|\}$. The set $\texttt{RPGG}_s(f_e)$ is equal to the set $PredsSyst(\mathcal{P}_{se} \cup \texttt{RPGG}_{se}(f_e))$. By refining with these predicates we ensure the monotonicity of the abstraction of the system's transition relation.

5.2 Refining the Abstract Observation Equivalence

If for every $\tau \in Traces(f_e)$ the formula $F(f_e, \tau)$ is satisfiable, the predicates from $\texttt{RPGG}(f_e)$ might not suffice to eliminate the counterexample, because the reason for its existence is the coarseness of the abstract observation-equivalence. We propose an algorithm RPOE (Refinement Predicates for the Observation Equivalence) for computing a set of observable refinement predicates that allow us to distinguish the concrete error paths for different traces. The predicates are obtained from interpolants for unsatisfiable conjunctions of trace formulas.

According to the construction of these formulas, they share only observable variables and hence the computed interpolants contain only observable predicates. The key challenge for the interpolation computation in our case is to ensure that these predicates are *localized*, i.e., that the variables which occur in an atom correspond to a single concrete state and not to a sequence of concrete states. We extend the algorithm from [10], which reduces the computation of interpolants for linear arithmetic to linear programming problems, in order to handle this additional condition on the variable occurrences. Our more general algorithm LILA (Linear Interpolation with Localized Atoms) receives in addition a partitioning of the variables which occur in the input systems of inequalities and as a result, each atom in the generated interpolant is guaranteed to contain variables from exactly one partition. We first present the algorithm RPOE and then describe the procedure LILA.

Algorithm:RPOE

Input: symbolic game $\mathcal{G} = (\mathcal{S}, err)$, pair $\mathcal{P} = (\mathcal{P}_{se}, \mathcal{P}_s)$ of finite sets of predicates,
strategy tree $T(f_e)$ of an abstract winning environment strategy in $\alpha(\mathcal{G}, \mathcal{P})$
Output: pair of sets of refinement predicates (R_{se}, R_s)
$\Phi := \{F(f_e, \tau) \mid \tau \in \mathit{Traces}(f_e)\}$; $R_{se} := \emptyset$;
while *all elements of* Φ *are satisfiable* **do**
  pick $\Psi \subseteq \Phi$, $\varphi \in \Phi \setminus \Psi$ **such that** $\psi := \bigwedge_{\phi \in \Psi} \phi$ **is satisfiable and** $\varphi \wedge \psi$ **is unsatisfiable;**
  $n := \mathit{max}(\mathit{MaxIx}(\varphi), \mathit{MaxIx}(\psi))$**;**
  if $R_{se} = \emptyset$ **then** $R_{se} := \mathit{OutPreds}(\{\tau \mid F(f_e, \tau) \in \{\varphi\} \cup \Psi\})$**;**
  $\theta := \mathtt{LILA}(\varphi, \psi, (\mathit{Vars}^0(\varphi) \cup \mathit{Vars}^0(\psi), \ldots, \mathit{Vars}^n(\varphi) \cup \mathit{Vars}^n(\psi)))$**;**
  $R_{se} := R_{se} \cup (\mathit{Preds}(\theta))\ \mathit{subst}_X$**;** $\Phi := \{\theta \wedge \phi \mid \phi \in \Psi\}$**;**
return $(R_{se}, \mathit{PredsSyst}(\mathcal{P}_{se} \cup R_{se}))$;

Distinguishing abstract prefixes. Let $\Phi = \{F(f_e, \tau) \mid \tau \in \mathit{Traces}(f_e)\}$. As all formulas $F(f_e, \tau)$ are satisfiable and the formula $F(f_e)$ is not, there exists a subset Ψ of Φ such that $\psi = \bigwedge_{\phi \in \Psi} \phi$ is satisfiable and there exists a formula $\varphi \in \Phi \setminus \Psi$ such that $\varphi \wedge \psi$ is unsatisfiable.

The variables in $\bigcup_{\pi \in T(f_e), \sigma_1, \sigma_2 \in C_o^*} X^{(\pi, \sigma_1, \sigma_2)}$ (and hence the variables in φ and in ψ) are partitioned according to the length of π: For $j \in \mathbb{N}$, X^j is the union of all sets $X^{(\pi, \sigma_1, \sigma_2)}$ with $|\pi| = j$. For a formula ϕ, $\mathit{MaxIx}(\phi)$ is the maximal j with $\mathit{Vars}^j(\phi) = \mathit{Vars}(\phi) \cap X^j \neq \emptyset$.

When φ and ψ are (disjunctions of) mixed systems of linear inequalities, we apply algorithm LILA described in the next paragraph to compute an interpolant θ such that each literal which occurs in θ is of the form $ix \lhd \delta$ where $\lhd \in \{\leq, <\}$ and the only variables which occur in such an inequality are in the set $\{x_i^{(\pi,\sigma)}, x_o^{(\pi,\sigma)}, t^{(\pi,\sigma)}\}$ for some $\pi \in T(f_e)$ and $\sigma \in C_o^*$, i.e. the coefficients in front of all other variables are 0. By applying the substitution subst_X to the atoms in θ, we obtain a set of predicates over observable variables from the original set of variables X. Then, the set Φ is updated to be the set of conjunctions $\theta \wedge \phi$, where $\phi \in \Psi$ and the process is repeated while all elements of the current set Φ are satisfiable. The predicates in $\mathtt{RPOE}_{se}(f_e)$ are the atoms from all computed interpolants, plus the set of output predicates for the traces corresponding to the formulas in the initial set $\Psi \cup \{\varphi\}$. The predicates in $\mathtt{RPOE}_s(f_e)$ ensure the monotonicity of the abstraction.

Computing interpolants with localized atoms. We now present the algorithm LILA for computing localized interpolants. A mixed system, denoted $Ax \leqslant a$, consists of strict and non-strict linear inequalities. The input of algorithm LI from [10] consists of two mixed

systems of inequalities $Ax \leqslant a$ and $Bx \leqslant b$ such that the conjunction $Ax \leqslant a \wedge Bx \leqslant b$ is not satisfiable. The output is a linear interpolant $ix \lhd \delta$ where $\lhd \in \{\leq, <\}$. Algorithm LILA receives in addition a partitioning $(V^0, V^1, \ldots, V^n)$ of the variables in the vector x. The output is an interpolant for $Ax \leqslant a$ and $Bx \leqslant b$ which is of the form $\bigwedge_{j=0}^{n} i_j x \lhd_j \delta_j$, where $\lhd_j \in \{\leq, <\}$ and for each $0 \leq j \leq n$, only variables from V^j occur in $i_j x \lhd_j \delta_j$. If such interpolant is not found, the element $\bot$ is returned. The variables $\lambda, \lambda_0, \lambda_1, \ldots, \lambda_n$ denote vectors which define linear combinations of inequalities in $Ax \leqslant a$. The subvectors $\lambda^{\mathsf{lt}}, \lambda^{\mathsf{le}}, \lambda_j^{\mathsf{lt}}, \lambda_j^{\mathsf{le}}$ for $j = 0, 1, \ldots, n$ define linear combinations of strict and non-strict inequalities in $Ax \leqslant a$, respectively. Similarly for $\mu, \mu^{\mathsf{lt}}, \mu^{\mathsf{le}}$. For each $0 \leq j \leq n$, the set of variables V^j defines a set $Ix(j)$ of indices: $Ix(j) = \{k \mid k \in \{1, \ldots, m_A\}, x_k \in V^j\}$, where m_A is the number of columns in A. Its complement $\{1, \ldots, m_A\} \setminus Ix(j)$ is denoted with $\overline{Ix(j)}$. For $1 \leq k \leq m_A$, the k-th column of the matrix A is denoted with $A_{|k}$. For disjunctions of mixed systems, i.e., for formulas $\bigvee_k A_k x \leqslant a_k$ and $\bigvee_l B_l x \leqslant b_l$ in DNF, we proceed as in [10]: compute an interpolant θ_{kl} for each pair of disjuncts and then take $\bigvee_k \bigwedge_l \theta_{kl}$.

THEOREM 3. *Algorithm* LILA *is* **sound***: If it returns a conjunction $\theta = \bigwedge_{j=0}^{n} \theta_j$, then θ is an interpolant for the pair of mixed systems $Ax \leqslant a$ and $Bx \leqslant b$ with the following properties: (1) for each j, θ_j is of the form $i_j x \lhd_j \delta_j$ where $\lhd_j \in \{\leq, <\}$; (2) there exist row vectors $\lambda_0, \ldots, \lambda_n$ such that for every $0 \leq j \leq n$, $\lambda_j \geq 0$, $i_j = \lambda_j A$ and $\delta_j = \lambda_j a$; (3) for each $0 \leq j \leq n$, only variables from V^j occur in θ_j. Algorithm* LILA *is* **complete***: if an interpolant $\theta = \bigwedge_{j=0}^{n} \theta_j$ with the properties (1),(2) and (3) exists, then the algorithm will find one.*

Algorithm:LILA
Input: $Ax \leqslant a$ and $Bx \leqslant b$: mixed systems, $Ax \leqslant a \wedge Bx \leqslant b$ is unsatisfiable, partitioning $(V^0, V^1, \ldots, V^n)$ of the variables in x
Output: interpolant $\bigwedge_{j=0}^{n} i_j x \lhd_j \delta_j$ where $\lhd_j \in \{\leq, <\}$ and only variables from V^j occur in $i_j x \lhd_j \delta_j$
$\chi_1 := \lambda \geq 0 \wedge \mu \geq 0 \wedge \lambda A + \mu B = 0$;
$\chi_2 := \lambda = \sum_{j=0}^{n} \lambda_j \wedge \bigwedge_{j=0}^{n} (\lambda_j \geq 0 \wedge i_j = \lambda_j A \wedge \delta_j = \lambda_j a \wedge \bigwedge_{k \in \overline{Ix(j)}} \lambda_j A_{|k} = 0)$;
if exist $\lambda, \mu, \lambda_j, i_j, \delta_j$, for $0 \leq j \leq n$ **satisfying** $\chi_1 \wedge \chi_2 \wedge \lambda a + \mu b \leq -1$
then return $\bigwedge_{j=0}^{n} i_j x \leq \delta_j$;
elif exist $\lambda, \mu, \lambda_j, i_j, \delta_j$, for $0 \leq j \leq n$ **satisfying** $\chi_1 \wedge \chi_2 \wedge \lambda a + \mu b \leq 0 \wedge \lambda^{\mathsf{lt}} \neq 0$
then return $\bigwedge_{0 \leq j \leq n, \lambda_j^{\mathsf{lt}} \neq 0} i_j x < \delta_j \wedge \bigwedge_{0 \leq j \leq n, \lambda_j^{\mathsf{lt}} = 0} i_j x \leq \delta_j$;
elif exist $\lambda, \mu, \lambda_j, i_j, \delta_j$, for $0 \leq j \leq n$ **satisfying** $\chi_1 \wedge \chi_2 \wedge \lambda a + \mu b \leq 0 \wedge \mu^{\mathsf{lt}} \neq 0$
then return $\bigwedge_{j=0}^{n} i_j x \leq \delta_j$;
else return $\bot$

5.3 Refinement Loop

In each iteration of the refinement loop, an abstract perfect-information game is solved. If it is won by the system player, the algorithm terminates returning an abstract winning strategy for the system. Otherwise, the abstraction is refined with the predicates $R(f_e)$, computed for some abstract winning strategy f_e for the environment. There are two cases. If refining the transition relations suffices to eliminate f_e, the abstraction is refined with the predicates in RPGG(f_e). Otherwise, the predicates in RPOE(f_e) are used for refinement. In

the second case, it is possible that in the game $\alpha(\mathcal{G}, \mathcal{P} \cup \texttt{RPOE}(f_e))$, the environment has a winning strategy f'_e with $f'_e \leq f_e$. Then, we also refine with the predicates in $\texttt{RPGG}(f'_e)$ for every such f'_e. The set $\mathit{Refine}(f_e, \mathcal{P}')$ consists of all winning strategies for the environment in $\alpha(\mathcal{G}, \mathcal{P}')$ subsumed by f_e. It can be computed from the strategy f_e and the predicates in $\mathcal{P}'$.

Algorithm:ARGII
Input: symbolic safety game $\mathcal{G} = (\mathcal{S}, \mathit{err})$ **Output:** pair $(\mathit{winner}, \mathit{abstract\ strategy})$
$\mathcal{P} := (\mathcal{P}_{se}, \mathcal{P}_s)$, **where** $\mathcal{P}_{se} := \mathit{Preds}(\mathit{init}) \cup \mathit{Preds}(\mathit{err}) \cup \{t \approx 0\}$ **and** $\mathcal{P}_s := \mathit{PredsSyst}(\mathcal{P}_{se})$;
solve $\alpha(\mathcal{G}, \mathcal{P})$ **and determine:** *winner* **and** *strategy*;
while *winner* = *env* **do**
  if $F(\mathit{strategy})$ **is satisfiable then return** $(\mathit{winner}, \mathit{strategy})$;
  $f_e := \mathit{strategy}$;
  if $\exists \tau \in \mathit{Traces}(f_e) : F(f_e, \tau)$ *is unsatisfiable* **then compute** $R := \texttt{RPGG}(f_e)$;
  else
    $R := \texttt{RPOE}(f_e)$; **compute** $S := \mathit{Refine}(f_e, R)$;
    forall $f'_e \in S$ **do** $R := R \cup \texttt{RPGG}(f'_e)$;
  $\mathcal{P} := \mathcal{P} \cup R$; **solve** $\alpha(\mathcal{G}, \mathcal{P})$ **and determine** *winner* **and** *strategy*;
return $(\mathit{winner}, \mathit{strategy})$

THEOREM 4. *[Soundness of algorithm* ARGII*] The algorithm* ARGII *is sound: if it returns (sys, f^a_s), then the concrete symbolic game $(\mathcal{S}, \mathit{err})$ is won by the system and f^a_s is a concretizable abstract winning strategy for the system; if it returns (env, f^a_e) then $(\mathcal{S}, \mathit{err})$ is won by the environment and f^a_e is a concretizable abstract winning strategy for the environment.*

THEOREM 5. *[Progress property of the refinement] Let f_e be a spurious wining strategy for the environment in the game $\alpha((\mathcal{S}, \mathit{err}), \mathcal{P})$. In $\alpha((\mathcal{S}, \mathit{err}), \mathcal{P} \cup R(f_e))$, the environment does not have a winning strategy f'_e with $f'_e \leq f_e$.*

6 Termination of the Abstraction Refinement Loop

In this section we provide sufficient conditions for termination of the refinement loop. In order to guarantee that only finitely many different abstract states are generated during the execution of the algorithm, we make standard assumptions about the concrete game graph, which we extend with conditions related to the presence of incomplete information. As we also have to account for the refinement predicates obtained from interpolants, we apply the standard technique (e.g, [8]) of restricting the interpolants computed at each step to some finite language $\mathcal{L}_b$ and maintaining completeness by gradually enlarging the restriction language when this is needed. We make use of the fact that our algorithm reduces interpolant computation to constraint solving, in order to achieve the restriction of the language by imposing additional constraints on the generated inequalities.

Computing restricted linear interpolants. We restrict the language of the computed interpolants to the set of rectangular predicates over the variables in $\mathit{Obs}(X)$. A *rectangular predicate* over $\mathit{Obs}(X)$ is a conjunction of *rectangular inequalities* of the form $ax \lhd c$, where $x \in \mathit{Obs}(X)$, $a \in \{-1, 1\}$, $\lhd \in \{<, \leq\}$ and c is an integer constant. For $m \in \mathbb{N}$, a rectangular predicate φ is called *m-bounded* if for each conjunct $ax \lhd c$ of φ, $|c| \leq m$. Let $\mathcal{L}_m$ be the set of all m-bounded rectangular predicates over $\mathit{Obs}(X)$. The modified algorithm $\texttt{LILA}_\texttt{r}$ gets

as input also a bound $b \in \mathbb{N}$ and ensures that every conjunct in the computed interpolant is in $\mathcal{L}_b$. If such an interpolant does not exist, then the bound b is increased. The modified algorithm partitions the variables into singleton sets and uses in conjunction with χ_1 and χ_2 the additional constraints: (1) χ_3 defined as $\chi_3 = \bigwedge_{j=0}^{n}(i_j \leq 1 \wedge i_j \geq -1 \wedge \delta_j \leq b \wedge \delta_j \geq -b)$ and (2) the variables δ_j and the variables in the vector i_j assume integer values.

Region algebra for an incomplete-information game. A *region algebra for a symbolic safety game* $(\mathcal{S}, err)$ is a pair (R, Obs) of possibly infinite sets $R \subseteq 2^S$ and $Obs \subseteq R$ of regions with the following properties: (1) for every $r_1, r_2 \in R$, we have $r_1 \cup r_2, r_1 \cap r_2, S \setminus r_1 \in R$; (2) for every $r_1, r_2 \in Obs$, we have $r_1 \cup r_2, r_1 \cap r_2, S \setminus r_1 \in Obs$; (3) the sets $\{v \in S \mid v \models t \approx 0\}$ and $\{v \in S \mid v \models t \approx 1\}$ are in R; (4) for every $r \in R$ and $c_o \in C_o$, and every $p \in Preds(Pre_e(r)) \cup Preds(Pre_s(c_o, r))$ it holds that for every $r' \in R$, either for every $v \in r'$, $v \models p$ or for every $v \in r'$, $v \models \neg p$; (5) for every $c_o \in C_o$, the set $\{v \in S \mid v \models x_o \approx c_o\}$ is in Obs; (6) for every $\pi_1, \pi_2 \in Prefs(G)$, if each of $last(\pi_1)$ and $last(\pi_2)$ is an error state and there exists an index j such that $\pi_1[j]$ and $\pi_2[j]$ are system states and $\pi_1[j] \not\equiv \pi_2[j]$, then there exist $0 \leq k \leq j$ and $r \in Obs$ such that $\pi_1[k] \in r$ and $\pi_2[k] \notin r$.

THEOREM 6. *[Termination] Consider a symbolic safety game $(\mathcal{S}, err)$ for which there exists a finite region algebra (R, Obs) with $Obs = \mathcal{L}_m$ for some $m \in \mathbb{N}$. If algorithm* `ARGII` *using the modified algorithm* $\texttt{LILA}_\texttt{r}$ *is called with argument $(\mathcal{S}, err)$, then it terminates.*

References

[1] T. Ball, B. Cook, S. Das, and S. K. Rajamani. Refining approximations in software predicate abstraction. In *TACAS*, volume 2988 of *LNCS*, pages 388–403. Springer, 2004.

[2] K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games of incomplete information. In *Proc. CSL*, volume 4207 of *LNCS*. 2006.

[3] S. Das and D. L. Dill. Counter-example based predicate discovery in predicate abstraction. In *Proc. FMCAD*, pages 19–32, London, UK, 2002. Springer-Verlag.

[4] L. de Alfaro and P. Roy. Solving games via three-valued abstraction refinement. In *Proc. CONCUR*, volume 4703, pages 74–89. Springer-Verlag, 2007.

[5] M. De Wulf, L. Doyen, and J.-F. Raskin. A lattice theory for solving games of imperfect information. In *Proc. HSCC*, LNCS, pages 153–168. Springer-Verlag, 2006.

[6] R. Dimitrova and B. Finkbeiner. Abstraction refinement for games with incomplete information. Reports of SFB/TR 14 AVACS 43, SFB/TR 14 AVACS, October 2008.

[7] T. Henzinger, R. Jhala, and R. Majumdar. Counterexample-guided control. In *Proc. ICALP'03*, volume 2719 of *LNCS*, pages 886–902. Springer-Verlag, 2003.

[8] R. Jhala and K. L. McMillan. A practical and complete approach to predicate refinement. In *Proc. TACAS*, volume 3920, pages 459–473. Springer-Verlag, 2006.

[9] J. H. Reif. The complexity of two-player games of incomplete information. *J. Comput. Syst. Sci.*, 29(2):274–301, 1984.

[10] A. Rybalchenko and V. Sofronie-Stokkermans. Constraint solving for interpolation. In *Proc. VMCAI*, volume 4349 of *LNCS*, pages 346–362. Springer-Verlag, 2007.