LTL falsification in infinite-state systems

Alessandro Cimatti, Alberto Griggio, Enrico Magnago*

Fondazione Bruno Kessler, Trento, Italy

Abstract

In infinite-state systems not all false LTL properties admit lasso-shaped witnesses. In this work, we propose an automatic approach that presents potentially non-lasso witnesses in an indirect way. The approach is based on two key insights. First, we define the notion of *well-founded funnel* to describe a source set of states that inevitably reach a destination set. We show that, under suitable conditions, a sequence of funnels ensures the existence of a fair path. Second, we adopt a compositional approach to partition the original system into projections and employ them together with funnels to build the indirect witness. Then, we propose an algorithm that identifies candidate funnels, proves their well-foundedness, and searches for a sequencing order. We experimentally evaluate the approach on examples taken from software, timed and hybrid systems, showing its wide applicability and expressiveness, with an implementation that outperforms various competitor tools.

Keywords: First-Order Linear-Time Temporal Logic, SMT-based Model Checking, Temporal Satisfiability, Infinite-State Transition Systems

1. Introduction

A well-known result in finite-state LTL model checking guarantees that the verification problem is decidable and, in particular, if a system does not satisfy a

☆This is a pre-print version of the homonymous paper published in Information and Computation [1]. It combines and extends the works presented in [2] and [3].

*Corresponding author

Email addresses: cimatti@fbk.eu (Alessandro Cimatti), griggio@fbk.eu (Alberto Griggio), magnago@fbk.eu (Enrico Magnago) property there exists a witness in the form of a lasso-shaped fair path [4]. Model checking of LTL properties in infinite-state systems (e.g. software programs, infinite-state transition systems, timed transition systems and hybrid systems) is an undecidable problem and there could be no lasso-shaped witness for the violation of some property.

A well-known instance of this problem is software (non)termination. In this context closed recurrence sets [5] are used to represent a witness for the nontermination of some software program. A closed recurrence set consists of a reachable set of states that is disjoint from the end states and inductive with respect to a left-total transition relation that underapproximates the transition relation of the program. The set represents at least one infinite execution for the program: (i) its reachability ensures that there is some finite execution of the program ending in some state within the set; (ii) since the set is also inductive, we know that no transition starting from within the set can reach a state outside of it and (iii) the left-total transition relation ensures that there always exists at least one successor state satisfying also the transition relation of the program.

In this work we are interested in representing *fair paths* of transition systems. Therefore, we do not look for any infinite execution, as in the nontermination case, but consider only those that visit a given set of states, called fair states, infinitely often. Recurrent sets are not sufficient, apart from some trivial cases, to conclude that every infinite execution visits some fair state infinitely often. Unless the set underapproximates the fair states, without additional information, we cannot conclude that the infinite executions described by the closed recurrence set are fair. For this reason, we split the closed recurrence set into two components S and D such that D is a subset of the fair states. The union of S and D must satisfy the same conditions described above for closed recurrence sets and, in addition, the left-total transition relation must not allow for infinite sequences of S states: every state in S must reach a state in D in a finite number of steps.

When writing or reasoning on a transition system, a human usually restricts its attention to a component at-a-time and partitions the state-space into regions such that all states in a region exhibit similar features and the system visits the regions in some order. We propose an approach that mimics this kind of reasoning by splitting the monolithical problem described above into two orthogonal directions: by *segmenting* the infinite paths into finite paths and *decomposing* the system with respect to some partitioning of the symbols.

We segment the fair paths into a concatenation of finite paths: we split S into multiple regions such that each region represents a set of finite paths that must eventually reach the following region. Notice that, while each path in a region must be finite, there might be no upper bound to their length: a region can represent an infinite number of finite paths with increasing lengths. We call each segment *funnel* and their concatenation representing the fair paths *funnel-loop*. In addition, we *decompose* the system by partitioning its symbols. Each component, called E-component (for existential component), describes the behaviour of a subset of the symbols while assuming some properties about the others. These properties represent the conditions that are necessary for this behaviour to be enabled and we need to prove that such conditions are ensured by some other component.

The main contributions of this work are the following: (i) we define an indirect representation of a non-empty set of fair paths for a transition system using funnel-loops; (ii) we show such representation to be both sound and relatively complete; (iii) we partition the search problem in two orthogonal directions: segmentation and decomposition; (iv) we define a search procedure capable of identifying funnel-loops; (v) finally, we show the wide applicability and effectiveness of the proposed procedure via a prototype implementation.

This work is an extension and an integration of our previous works presented in [2] and [3]. In this article, we unify the two approaches in an integrated framework, in which the search for a funnel-loop witnessing the falsification of a given property (first introduced in [3]) can be decomposed by using the E-component concept of [2] extended with ranking functions.[1] Moreover, we

[1] E-components (without ranking functions) were called AG-skeletons in [2].

enrich the results of [3] by a relative-completeness theorem for the representation of fair paths as funnel-loops, and an encoding for the search problem of a funnel-loop in existentially-quantified constrained Horn clauses. Finally, we provide all the proofs of our results and we revised our experimental evaluation considering also an additional competitor tool.[2]

The paper is structured as follows. In Section 2 we describe the notation and introduce the constructs we use in the following sections. In Section 3 we provide an overview of the proposed approach. In Section 4 we introduce a running example that we will use to illustrate our procedures. In Section 5 we define funnels and funnel-loops, prove their properties and show how they can be used to identify fair paths for a fair transition system. In Section 6 we define E-components, their composition and projection operators and show the relationship between E-components and funnel-loops. In Section 7 we present an algorithm to search for a funnel-loop describing a non-empty set of fair paths of a fair transition system. In Section 8 we discuss the related work. In Section 9 we briefly describe some implementation details of our prototype and then discuss our experimental results. In Section 10 we draw some conclusions and outline the directions for future work.

2. Background

We work in the setting of SMT, with the theory of quantified mixed integer-real nonlinear arithmetic. We assume the standard notions of interpretation, model, satisfiability, validity and logical consequence.

2.1. Symbols, formulae, implicants and entailment

Given a set of symbols V, we use $V' \doteq \{v' | v \in V\}$ for the set containing the primed version of the symbols. We write $\phi(V)$ for a Boolean formula over the symbols in V and $\phi(V, V')$ for a Boolean formula or relation over $V \cup V'$. When clear from the context we will omit the set of symbols and simply write

[2]In order to aid readability, some of the more technical proofs are presented in appendix.

ϕ, ψ and ϕ' for $\phi(V)$, $\psi(V,V')$ and $\phi(V')$ respectively. We say that a formula $\phi(V,V')$ underapproximates a formula $\psi(V,V')$ iff every time ϕ holds then also ψ must hold, hence $\phi \rightarrow \psi$ is valid. We use $\top$ and $\bot$ in formulae to represent respectively the true and false Boolean constants.

We denote with $\boldsymbol{v}$ a total assignment over V, i.e. a state. Given a formula $\phi(V)$ we write $\phi(\boldsymbol{v})$ for the evaluation of ϕ obtained by replacing every symbol in V with its corresponding assignment in $\boldsymbol{v}$ and $\phi(\boldsymbol{v}')$ for the evaluation of ϕ where every symbol $v \in V$ is replaced by the assignment of v' in $\boldsymbol{v}'$. We overload the $\models$ symbol: when ϕ and ψ are SMT formulae, then $\phi \models \psi$ stands for entailment in SMT; when M is a fair transition system and ψ is a linear temporal property, then $M \models \psi$ is to be interpreted with the LTL semantics.

Finally, if ψ is a quantifier-free SMT formula and ϕ is a conjunction of (a subset of) the atoms of ψ, then ϕ is an implicant of ψ iff $\phi \models \psi$.

2.2. Well-founded relations and ranking functions

A binary relation $\rho \subseteq Q \times Q$ is well-founded if every non-empty subset $U \subseteq Q$ has a minimal element with respect to ρ, i.e. there is $m \in U$ such that no $u \in U$ satisfies $\rho(u,m)$. Given a relation $\phi(V,V')$, a ranking function $\textsc{Rf}(V)$ is a function from the assignments to the symbols V to some set Q, such that the relation $< \doteq \{\langle \textsc{Rf}(\boldsymbol{v_0}), \textsc{Rf}(\boldsymbol{v_1'})\rangle \mid \boldsymbol{v_0}, \boldsymbol{v_1'} \models \phi\}$ is well-founded and we call $\mathbf{0}$ its minimal element. Given a set of ranking functions $\{\textsc{Rf}_i\}_{i=0}^{n}$, we define their sum as $\textsc{Rf} \doteq \sum_{i=0}^{n} \textsc{Rf}_i \doteq \langle \textsc{Rf}_0, \ldots, \textsc{Rf}_n\rangle$. $\textsc{Rf}$ is a ranking function with minimal element $\mathbf{0} \doteq \langle \mathbf{0}_0, \ldots, \mathbf{0}_n\rangle$ and comparison operator $< \doteq \{\langle \boldsymbol{v}_0, \boldsymbol{v}_1\rangle | (\bigwedge_{i=0}^{n} \textsc{Rf}_i(\boldsymbol{v}_0) \leq_i \textsc{Rf}_i(\boldsymbol{v}_1)) \wedge (\bigvee_{i=0}^{n} \textsc{Rf}_i(\boldsymbol{v}_0) <_i \textsc{Rf}_i(\boldsymbol{v}_1))\}$ where $<_i$ is the well-founded relation associated with $\textsc{Rf}_i$ and $\leq_i$ is a shortcut for the disjunction of $<_i$ and the equality.

2.3. LTL *model checking*

A symbolic fair transition system M is a tuple $\langle V, I(V), T(V,V'), F(V)\rangle$, where V is the set of state variables; I and F denote respectively the initial and fair states; and T represents the transitions where V' refers to the next state variables. A path or trace of M is a finite or infinite sequence of states $\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots$, such that $\boldsymbol{v}_0 \models I$ and $\boldsymbol{v}_i, \boldsymbol{v}'_{i+1} \models T$ for all i, where $\boldsymbol{v}'_i$ assigns to every symbol $v' \in V'$ the same value assigned by $\boldsymbol{v}_i$ to v. A state $\boldsymbol{v}$ is reachable in M if there is a finite path of M ending in $\boldsymbol{v}$. Given a formula $\phi(V)$ we write $M \rightsquigarrow \phi$ iff there exists a finite path in M ending in a state $\boldsymbol{v}$ such that $\boldsymbol{v} \models \phi$.

A path $\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots$ of M is fair iff for each i there exists $j > i$ such that $\boldsymbol{v}_j \models F$ and the language of M, written $\mathcal{L}(M)$, is the set of all fair paths of M.[3] We also assume the standard notions of temporal logic model checking, using the usual definitions of $\mathbf{U}, \mathbf{G}, \mathbf{F}$ for the "until", "always" and "eventually" temporal operators (LTL [6]): for a LTL property φ we write $M \models \varphi$ iff φ holds in every path $\pi \in \mathcal{L}(M)$. Given a fair transition system M, we are interested in the problem of deciding whether M admits at least one fair path (i.e. $\mathcal{L}(M) \neq \emptyset$). Notice that the existential LTL model checking problem, i.e. the problem of deciding whether a system $M \doteq \langle V, I, T, \top \rangle$ admits at least a path that satisfies a given LTL formula φ, can be reduced to checking for the existence of a fair path in the fair transition system $M \times M_\varphi \doteq \langle V \cup V_\varphi, I \wedge I_\varphi, T \wedge T_\varphi, F_\varphi \rangle$, where $M_\varphi \doteq \langle V_\varphi, I_\varphi, T_\varphi, F_\varphi \rangle$ is a symbolic encoding of an automaton accepting the language of φ [7], which can be obtained, for example, with the procedure of [8]. In addition, in finite-state systems liveness-to-safety [9] allows the reduction of such problem to the falsification of safety properties.

A standard technique for the analysis of infinite-state systems is predicate abstraction [10]. Predicate abstraction partitions the state space according to the equivalence relation induced by a set of predicates. Given a finite set of predicates, it defines a finite set of abstract states each of which corresponds to a total truth assignment of such predicates. An abstract state corresponds to a possibly infinite set of concrete states: all states that agree on the truth assignments of the predicates. Implicit abstraction is an approach to avoid the explicit computation of the abstract space. Implicit abstraction has been used, e.g. in [11], in combination with liveness-to-safety to identify abstract fair loops

[3]Note that fair paths are necessarily infinite.

for an infinite-state system in the abstract space. However, in general, there might not exist a fair path in the concrete system corresponding to the abstract one.

Finally, we consider also timed systems such as timed automata [12], timed transition systems [13] and hybrid systems [14]. Timed systems are infinite-state transition systems in which each state is associated with a real-valued time and transitions may cause time to elapse. An LTL property holds in such systems iff it holds in all its *non-Zeno* paths. A path is *non-Zeno* iff the sequence of time points associated to its states is diverging (i.e. there is no upper bound on the value of time along the path).[4]

3. Overview of the approach

Our objective is to define a representation of and a search procedure for a non-empty set of fair paths for a transition system. We split the representation of the fair paths along two orthogonal directions. We first *segment* them into finite sequences of elements each of which represents a set of finite paths. Then, we *decompose* the fair transition system with respect to a partitioning of its symbols; in this case each component represents a set of infinite behaviours for a subset of the symbols. Therefore, the search problem is reduced to the problem of identifying an appropriate set of components such that their composition is a witness for some fair path in the transition system.

3.1. Segmentation: funnels

We *segment* fair paths into a sequence of elements called *funnels* that, like actual funnels, take items from a source and constrain them to follow a path leading to a destination. Funnels are compact witnesses for universal and existential reachability [15]: each funnel characterizes a set of finite paths, each starting from the source region, remaining in it for a bounded number of steps, and eventually ending in the destination region. Funnels are concatenated in

[4]As for fair paths, also non-Zeno paths are necessarily infinite.

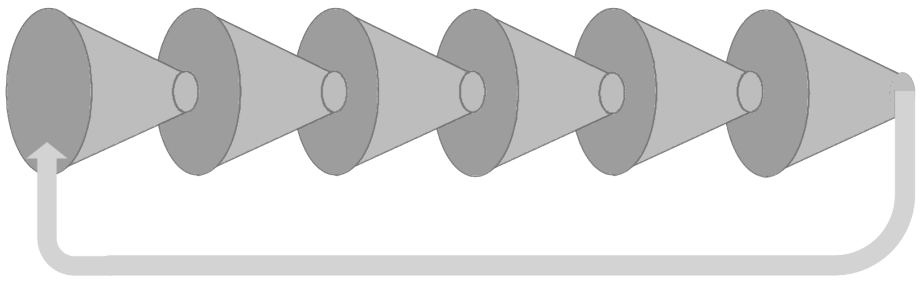

Figure 1: Funnels combined into chain forming a funnel-loop.

chains such that the destination region of a funnel is contained in the source region of the following one. Funnel-loops are chains of funnels in which the destination region of the last funnel is included in the source region of first one. An example of funnel-loop composed of 6 funnels is depicted in Fig. 1. Funnel-loops describe a loop over the regions of the corresponding funnels and we ensure the fairness of such loop by requiring at least one of the destination regions to contain only fair states. Therefore, we propose to represent witnesses for fair paths of transition systems by composing a finite number of reachability witnesses. Sec. 5 formally presents funnel-loops and shows them to be a sound (Th. 1) and complete (Th. 2) representation of fair paths.

3.2. Decomposition: existential components

We *decompose* a fair transition system with respect to a partitioning of its symbols. For each subset of the symbols, we identify components, called E-components (for *existential components*), that represent their behaviour with respect to a sequence of regions. E-components distinguish three kinds of transitions between their regions and all states in the same region must exhibit transitions of the same kind. In this sense, the regions of an E-component group states with similar behaviour. We define two operators over these structures. The first operation, called *projection*, shrinks the set of paths described by an E-component by considering only a subset of its regions. The second operation, called *composition*, defines how E-components can be composed to obtain a description of the behaviour of a larger set of symbols: the union of the symbols of the composed elements. In this setting we represent fair paths as the composition and projection of a finite set of E-components.

Sec. 6 formally defines E-components and the two operators, while Theorems 3 and 6 highlight the relationship between funnel-loops and E-components. In more detail, Th. 3 shows that a funnel-loop also defines a corresponding E-component and, viceversa, Th. 6 details the conditions under which an E-component corresponds to a funnel-loop proving the existence of at least one fair path for some fair transition system.

3.3. Search procedure

We propose a fully-automated procedure that, given a fair transition system and a possibly empty set of E-components, searches for a funnel-loop containing at least one and only fair paths (Alg. 1). We propose to search for a funnel-loop by enumerating candidate fair loops of the transition system (Alg. 2). We consider loops such that the first and last state of the path are in the same abstract state with respect to a set of abstraction predicates. For every such candidate loop we compute a sequence of regions and transitions containing it (Alg. 3). Then, we search for a funnel-loop corresponding to a strengthening of the sequence of regions and transitions such that all required hypotheses are met (Sec. 7.4). If the search succeeds we return the obtained funnel-loop, otherwise we continue by analysing the next candidate fair loop.

The procedure, described in Sec. 7, is fully-automated and deals with an undecidable problem. Therefore, there will always exist some inputs for which it fails to provide an answer and, from a more practical perspective, inputs for which it takes a very long time to provide an answer. For this reason, the procedure is capable of exploiting some additional information in the form of a set of E-components. If some E-components are provided, the procedure identifies candidate fair loops that are also a path for some composition and projection of a subset of the E-components. It then tries to identify the funnel-loop that corresponds to an E-component describing the behaviour for the missing symbols: it completes the E-component with a transition relation over the remaining symbols such that all assumptions are met.

4. Running example

We now introduce a simple LTL verification problem on a software program that will be used as running example throughout this work.

Consider the simple program shown in Fig. 2, where NONDET is a function that nondeterministically selects a value from the set provided as input. Our

```
1: **int** $x \leftarrow$ NONDET($\mathbb{Z}$)
2: **real** $y \leftarrow$ NONDET($\mathbb{R}$)
3: **while** $x^2 \geq xy$ **do**
4:     $y \leftarrow$ NONDET($\mathbb{R}$)
5:     $x \leftarrow x + 1$
6: **end while**
```

Figure 2: Running example.

objective is to check whether in every infinite execution of such program the value of y will eventually remain always positive or always negative. This statement can be written in LTL as $(\mathbf{FG} y \geq 0) \vee (\mathbf{FG} y \leq 0)$. Intuitively, any counterexample to such specification must be a nonterminating execution of the program in which both $y > 0$ and $y < 0$ hold infinitely often.

We encode the software program as an infinite-state transition system using an additional variable *pc* to model the program counter. Then, we employ the reduction from LTL model checking to the existence of a fair path. The resulting infinite-state transition system is $Ex \doteq \langle \{x, y, pc, f_0, f_1\}, pc = 3, T, f_0 \wedge f_1 \rangle$, where *pc* and x are two integer variables, y is a real variable, f_0 and f_1 are two Boolean symbols (introduced by the reduction to keep track of the fairness conditions

$y > 0$ and $y < 0$) and the transition relation is defined as follows:

$$\begin{aligned} T \doteq\ & (pc = 3 \rightarrow (x^2 \geq xy \wedge pc' = 4 \wedge x' = x \wedge y' = y)) \wedge \\ & (pc = 4 \rightarrow (pc' = 5 \wedge x' = x)) \wedge \\ & (pc = 5 \rightarrow (pc' = 3 \wedge x' = x + 1 \wedge y' = y)) \wedge \\ & ((f_0 \wedge f_1) \rightarrow (\neg f_0' \wedge \neg f_1')) \wedge \\ & (f_0' \rightarrow (f_0 \vee y > 0)) \ \wedge \ (f_1' \rightarrow (f_1 \vee y < 0)). \end{aligned}$$

The first three lines encode the transition relation of the program. Notice that every state such that $pc = 3$ and $\neg(x^2 \geq xy)$ hold is a deadlock for Ex. In all such cases, the transition relation admits no successor state. Finally, the last two lines of the formula ensure that in every execution in which $f_0 \wedge f_1$ holds infinitely often also $y > 0$ and $y < 0$ hold infinitely often.

5. Segmenting paths with funnels

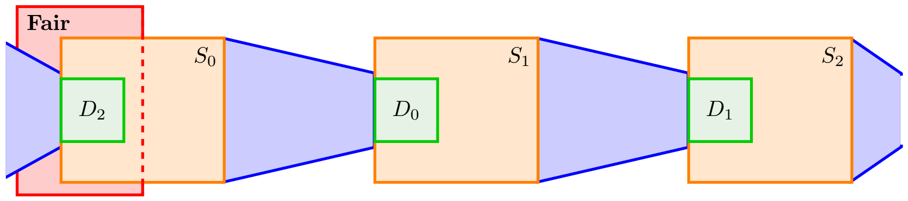

Figure 3: Funnels combined into chain forming a funnel-loop.

In this section, first we formally define *funnels* and their concatenation into funnel-loops (Fig. 3); then we provide a set of sufficient conditions for a funnel-loop to represent at least one fair path of a transition system and show that if such a fair path exists then also a corresponding funnel-loop must exist.

5.1. Funnels

Funnels *segment* fair paths into finite subpaths. Given a set of symbols V, a funnel is a 4-tuple $\langle S(V), T(V, V'), D(V), \textsc{Rf}(V)\rangle$. S and D are formulae representing respectively the source and destination regions, T is the transition relation and $\textsc{Rf}$ is a ranking function for S with respect to the transition relation T. Intuitively, this structure represents a terminating loop over S where D are the end states of the loop. Depending on the shape of the ranking function, the loop might correspond to a simple loop or to more complex termination arguments such as nested loops. Every path through the funnel starts from a state in S and follows the relation T such that it remains in S while the ranking function $\textsc{Rf}$ is greater than the minimal element $\mathbf{0}$ and, finally, it reaches a state in D when $\textsc{Rf}$ is $\mathbf{0}$. If we consider a trivial ranking function that is always equal to the minimal element $\mathbf{0}$ the 4-tuple simply asserts that every state in S is mapped into D by a single transition T.

Definition 1 (Funnel). *Given a set of symbols V, a funnel is defined as the 4-tuple*

$$fnl \doteq \langle S(V), T(V, V'), D(V), \textsc{Rf}(V)\rangle$$

where: $\textsc{Rf}$ *is a ranking function with minimal element* $\mathbf{0}$ *and S, D and T are formulae representing respectively the source region, destination region and transition relation of fnl. Every funnel satisfies the following hypotheses.*

F.1 *The transition relation is left-total relative to the source region.*

$$\forall V \exists V' : S \rightarrow T$$

F.2 *Every funnel keeps iterating on the source region as long as its ranking function is greater than the minimal element.*

$$\forall V, V' : (S \wedge \mathbf{0} < \textsc{Rf} \wedge T) \rightarrow S'$$

F.3 *Every step from the source region decreases the ranking function.*

$$\forall V, V' : (S \wedge \mathbf{0} < \textsc{Rf} \wedge T) \rightarrow \textsc{Rf}' < \textsc{Rf}$$

F.4 *Once the ranking function is equal to* $\mathbf{0}$ *the funnel reaches its destination region.*

$$\forall V, V' : (S \wedge \textsc{Rf} = \mathbf{0} \wedge T) \rightarrow D'$$

Given a funnel fnl_i we write S_i, T_i, D_i and $\textsc{Rf}_i$ to refer to its components. We define the transition system corresponding to a funnel $fnl \doteq \langle S, T, D, \textsc{Rf} \rangle$ over symbols V as $M_{fnl} \doteq \langle V, S, (\neg D \wedge T) \vee (D \wedge D'), \top \rangle$. We refer to the paths through a funnel fnl meaning the finite paths of the corresponding transition system that end in D and write $fnl \models \phi$ meaning that ϕ holds in every path in $\mathcal{L}(M_{fnl})$. Notice that the paths through a funnel are all finite and each of them is a prefix of some path in $\mathcal{L}(M_{fnl})$. From the definition it easily follows that every funnel fnl satisfies the following:

$$fnl \models S \ \mathbf{U} \ D$$

5.2. Funnel-loops

We define a funnel-loop as a chain of funnels $[fnl_i]_{i=0}^{n-1}$ such that the destination region of each funnel is included in the source region of the following one and the destination region of the last funnel is included in the source region of the first one.

Definition 2 (Funnel-loop). *A sequence of $n \geq 1$ funnels $[fnl_i]_{i=0}^{n-1}$ over symbols V is a funnel-loop iff the following hold.*

FL.1 *The destination region of a funnel is included in the source region of the following funnel.*

$$\forall 0 \leq i < n-1, V : D_i \rightarrow S_{i+1}$$

FL.2 *The destination region of the last funnel D_{n-1} is contained in the source region of the first funnel S_0.*

$$\forall V : D_{n-1} \rightarrow S_0$$

We define the paths through a funnel-loop *floop*, $\mathcal{L}(floop)$, as the infinite paths obtained by infinite concatenation of the paths through the funnels in the corresponding chain and write $floop \models \phi$ meaning that ϕ holds in all such paths. For every funnel different from the last one, Hyp. FL.1 ensures that we can extend every path of such funnel, ending in its destination region, by following the transition relation of the next funnel. Therefore, every path starting in any source region will eventually reach the destination region of the last funnel:

$$\mathit{floop} \models (\bigvee_{i=0}^{n-1} S_i) \ \mathbf{U} \ D_{n-1}$$

By Hyp. FL.2 every time we reach the destination region of the last funnel associated with floop we are also in the source region of the first funnel. Therefore, we can extend the execution by appending another finite number of steps: a finite path starting from S_0 and ending in the last destination region D_{n-1}. We can do this infinitely many times obtaining infinite paths.

$$\mathit{floop} \models \mathbf{G}((\bigvee_{i=0}^{n-1} S_i) \ \mathbf{U} \ D_{n-1})$$

The definition of funnel-loop allows for regions with non-empty intersections. This eases the construction of the structure in practical cases. It is possible to consider one funnel at a time and then chain them simply by checking the inclusion of each destination into the corresponding source region. However, for every funnel-loop there exists one with pairwise-disjoint regions that has the same language projected over the common variables.[5] For this reason, when proving statements about the language of these structures, we assume without loss of generality that the regions of every funnel in the funnel-loops are pairwise-disjoint.

We propose to identify a non-empty set of fair paths for a transition system M as a funnel-loop floop; every path through floop must correspond to an infinite fair execution of M. The totality of the transition relation of each funnel (F.1) and their chaining (FL.1, FL.2) ensure that all the paths in $\mathcal{L}(\mathit{floop})$ are infinite. We need such paths to be fair paths, hence they must visit the fairness condition infinitely often. By construction of floop we know that every path goes through each S_i and each D_i infinitely many times. Since by FL.1 and FL.2

[5]See Appendix B.1 for a proof.

for every source region S_i, there exists a destination region D_j that is contained in it, it is sufficient to require one of the destination regions to contain only fair states. Without loss of generality we assume such a region to be the last one. These conditions ensure that $floop$ represents a set of fair paths of M. However, such set might be empty or non-reachable in M. Therefore, we finally require the union of the source regions to contain at least one state reachable in M. The existence of such state is sufficient to conclude non-emptiness of $\mathcal{L}(floop)$ because the transition relation of each funnel always allows for a successor state (F.1) and, by induction, this ensures that every region and the language of $floop$ are not empty. Th. 1 shows that these requirements are sufficient for a funnel-loop to prove the existence of a fair path in M and Th. 2 shows that if M admits a fair path then there exists a funnel-loop of length one for M. Therefore, funnel-loops composed of a single funnel are expressive enough to represent any fair path. However, funnel-loops of greater length lead to a description easier to understand for a person and, in addition, could simplify the search procedure: we might not need to consider complex disjunctive representations of the regions, ranking functions and transition relations.

Theorem 1. *Let $M \doteq \langle V, I^M, T^M, F^M \rangle$ be a fair transition system. Let $floop$ be a funnel-loop of length n over the symbols V and funnels $[fnl_i]_{i=0}^{n-1}$ such that:*

FF.1 *There is at least one state in the union of the source regions of $floop$ that is reachable in M:*

$$M \rightsquigarrow \bigvee_{i=0}^{n-1} S_i$$

FF.2 *The destination region of the last funnel contains only fair states of M.*

$$\forall V \ : \ D_{n-1} \rightarrow F^M$$

FF.3 *Every transition of every funnel underapproximates the transition relation of M. For every funnel fnl_i in $[fnl_i]_{i=0}^{n-1}$:*

$$\forall V, V' \ : \ S_i \wedge T_i \rightarrow T^M$$

Then M admits at least one fair path.

Proof. We first prove that every path in $\mathcal{L}(floop)$ is infinite. Then we prove that every such path is fair with respect to the fairness condition F^M and that every step in every such path satisfies the transition relation T^M. Finally, we prove that $\mathcal{L}(floop)$ allows for at least one path which is a suffix of some path of M.

- *Every path in $\mathcal{L}(floop)$ is infinite.* Consider a funnel $fnl \doteq \langle S, T, D, \textsc{Rf} \rangle$ in $floop$. Hyp. F.1 ensures that its transition relation T allows for a successor state for every state in S. Hyp. F.2 ensures that every path of fnl remains in S while $\mathbf{0} < \textsc{Rf}$. Hyp. F.3 ensures that every such path will eventually reach a state in $S \wedge \textsc{Rf} = \mathbf{0}$. Hyp. F.4 ensures that every state in such region in one T step reaches a state in D. Therefore, every path starting from the source region S of each funnel can be extended until it reaches its destination region D. If fnl_{i-1} has a successor fnl_i in $floop$, by Hyp. FL.1 the destination region D_{i-1} is included in S_i: every state in D_{i-1} is also in S_i. Therefore, the concatenation of fnl_{i-1} and fnl_i allows to extend every path starting from either S_{i-1} or S_i until it reaches D_i. By induction this shows that the funnel chain allows the extension of every path starting from the union of the source regions until it reaches the last destination region:

  $$floop \models (\bigvee_{i=0}^{n-1} S_i) \;\mathbf{U}\; D_{n-1}$$

  Hyp. FL.2 requires the last destination region D_{n-1} to be a subset of the first source region S_0. As stated above, we can extend every path starting in every region until it reaches D_{n-1}, hence from S_0 we reach D_{n-1} again in a finite number of steps and at least one. Therefore, since we can extend each path of a finite non-zero number of steps infinitely many times every path in $\mathcal{L}(floop)$ is infinite.

- *Every path in $\mathcal{L}(floop)$ visits F^M infinitely often.* Hyp. FF.2 ensures that D_{n-1} underapproximates the fair states F^M. We have already shown above that every path of $floop$ reaches a state in D_{n-1} infinitely often. Therefore, such paths visit F^M infinitely often.

- *Every step of every path in $\mathcal{L}(floop)$ satisfies T^M.* Every step of every path in $\mathcal{L}(floop)$, by definition, corresponds to a transition of some funnel fnl. By hypotheses F.2, F.4, FL.1 and FL.2 every such path remains within the union of the regions and visits them following the order of the funnels. Therefore, every transition in every path of $floop$ must satisfy $S \wedge T$ for some funnel fnl in the sequence. Hyp. FF.3 ensures that if $S \wedge T$ holds that also T^M is true. Therefore every step of every path of $floop$ is also a step of M.

- *$\mathcal{L}(floop)$ allows for at least one path which is a suffix of some path of M.* Hyp. FF.1 ensures that there exists a finite path π_{pref} of M starting in I^M and ending in some state $\boldsymbol{v}$ such that $\boldsymbol{v} \models \bigvee_{i=0}^{n-1} S_i$. Therefore, $\boldsymbol{v}$ must be in S_i for some $0 \leq i < n$. Then, in $floop$ we can extend $\boldsymbol{v}$ to an infinite fair path π_{suf} starting in $\boldsymbol{v}$. As shown above every step of π_{suf} satisfies the transition relation of M and visits the fairness condition F^M infinitely often. The concatenation π of π_{pref} and π_{suf} without repetition of $\boldsymbol{v}$, starts from a state in I^M, every steps satisfies T^M and visits F^M infinitely often. Therefore, π is a fair path for M: $\pi \in \mathcal{L}(M)$.

□

Th. 2 ensures that if a transition system admits a fair path then there exists a corresponding funnel-loop, provided it is possible to represent the states in the path as formulae and, in particular, we are interested in finite formulae. In finite-state systems this is always the case: every set of states is finite and can be represented as a finite quantifier-free formula (e.g. the disjunction of the assignments in the set). However, this might not be the case in infinite-state systems: there might be an infinite set of states which cannot be represented by a finite formula. Therefore, the following theorem guarantees completeness relative to the expressiveness of the logic used to represent the regions and the transition relation of the funnel. Notice that the existence of a finite representation is not the only source of incompleteness. In fact, the existence of a finite formula does not imply the existence of a complete procedure capable of finding it. We remark that we are dealing with an undecidable problem, hence there exists no procedure to solve it that is both sound and complete.

Theorem 2. *If a fair transition system M admits at least one fair path, then there exists a funnel-loop $floop$ of length 1 for M. However, the existence of one representable via finite formulae depends on the expressiveness of the considered logic.*

Proof. In the following we will define a predicate $\phi(V)$ as the set of assignments $\boldsymbol{v}$ such that $\boldsymbol{v} \models \phi$, meaning that $\phi(V)$ is a formula equivalent to the disjunction of the assignments in the set. Notice that there might be no finite representation of ϕ.

Let $M \doteq \langle V, I^M, T^M, F^M \rangle$ and, by hypothesis, there exists a fair path π in $\mathcal{L}(M)$. Without loss of generality we assume that π visits every state at most once. If this is not the case, to obtain a fair path satisfying the hypothesis, it is sufficient to add an additional integer symbol whose assignment increases by one at every transition. In more detail, consider the fair transition system $\langle V \cup c, I^M \wedge c = 0, T^M \wedge c' = c + 1, F^M \rangle$, if π is a fair path of M then, we can obtain a fair path for the modified system by extending the assignment of every state of π such that $c = 0$ in the first state and in all other states assign to c the assignment of the previous state plus 1.

Let $floop$ be a funnel-loop of length 1, and let its funnel be $fnl \doteq \langle S, T, D, \textsc{Rf} \rangle$. We define the components of fnl as follows:

- S contains all and only states of π.

$$S \doteq \{\boldsymbol{v} \mid \boldsymbol{v} \in \pi\}$$

- D contains all and only the fair states of π.

$$D \doteq \{\boldsymbol{v} \mid \boldsymbol{v} \in \pi \wedge F^M(\boldsymbol{v})\}$$

- T is a relation containing all pairs of state $\langle \boldsymbol{v}, \boldsymbol{v}' \rangle$ such that $\boldsymbol{v}'$ is the successor state of $\boldsymbol{v}$ in π.

$$T \doteq \{\langle \boldsymbol{v}, \boldsymbol{v}' \rangle \mid \langle \boldsymbol{v}, \boldsymbol{v}' \rangle \in \pi\}$$

- $\textsc{Rf}$ associates to every state in π the number of steps required to reach the next fair state in π minus 1.

$$\forall k > 0, \forall V_1, \ldots, V_k : \textsc{Rf}(V_1) = k - 1 \leftrightarrow (\bigwedge_{i=1}^{k-1} T(V_i, V_{i+1})) \to F^M(V_k)$$

  This is well-defined since each state appears only once in π and by construction T allows for a single successor for each state. In addition, π is a fair path by hypothesis, hence there can be at most a finite number of non-fair states between every pair of fair states.

We now show that fnl satisfies all hypotheses of Def. 1.

F.1 π is an infinite sequence of states, all states of π are in S and each pair of subsequent states of π is in T. Therefore, T must be left-total with respect to S and Hyp. F.1 holds.

F.2 By construction S contains all states of π and T is a relation between states of π. Therefore, S is an inductive invariant for T and Hyp. F.2 holds.

F.3 By construction, $\textsc{Rf}$ is greater than 0 in all states that require more than 1 transition to reach a fair state and T is such that it brings all such states 1 step closer to the next fair state in π. Therefore, $S(V) \land 0 < \textsc{Rf}(V) \land T(V, V') \to \textsc{Rf}(V) = \textsc{Rf}(V') + 1$, which implies Hyp. F.3.

F.4 By construction $\textsc{Rf}$ assigns the minimal value 0 to the states that reach a fair state in 1 step. Therefore, Hyp. F.4 holds.

We now show that fnl corresponds to a funnel-loop $floop$ of length one: it satisfies all hypotheses of Def. 2.

FL.1 $floop$ contains a single funnel, hence Hyp. FL.1 trivially holds.

FL.2 By construction S contains all states of π while D contains the subset of states of π that are also fair. Therefore, $D \rightarrow S$ is valid and Hyp. FL.2 holds.

Finally, $floop$ represents fair paths of M: it satisfies all hypotheses of Th. 1

FF.1 π is a path of M, hence its first state is an initial state of M. All states of π are in S. Therefore, S contains at least 1 initial state of M and Hyp. FF.1 holds.

FF.2 The last destination region of $floop$ is D. By construction D contains only fair states, hence Hyp. FF.2 holds.

FF.3 π is a path of M. Therefore, every pair of states $\langle \boldsymbol{v}, \boldsymbol{v}' \rangle$ such that $\boldsymbol{v}'$ is the successor state of $\boldsymbol{v}$ in π, must also be in the relation T^M. By construction T contains only such pairs, hence $T \rightarrow T^M$ is valid and Hyp. FF.3 holds.

□

5.3. Example

We now define two funnel-loops, of length respectively 6 and 1, for the running example introduced in Sec. 4. Both funnel-loops are sufficient to conclude the existence of a fair path for the fair transition system Ex we defined in Sec. 4. Here we simply recall that the system has 5 state variables $V \doteq \{x, y, pc, f_0, f_1\}$ and that the fair states are all the states where $f_0 \wedge f_1$ holds.

We first describe the funnel-loop $floop \doteq [fnl_i]_{i=0}^{5}$ depicted in Fig. 4. The figure reports the source regions and transition relations of each funnel. The transitions in the figure report only the constraints for x and y, while the ones for pc, f_0 and f_1 can be trivially inferred by the assignments in the regions. More formally, each funnel fnl_i is the tuple $\langle S_i, T_i, D_i, \text{RF}_i \rangle$. We define each ranking function such that it is always equal to its minimal element, $\forall V : \text{RF}_i(V) = \mathbf{0}$, and each destination region as the corresponding source region, $D_i \doteq S_{(i+1)\%6}$. We define the remaining components, source regions and transition relations, as follows.

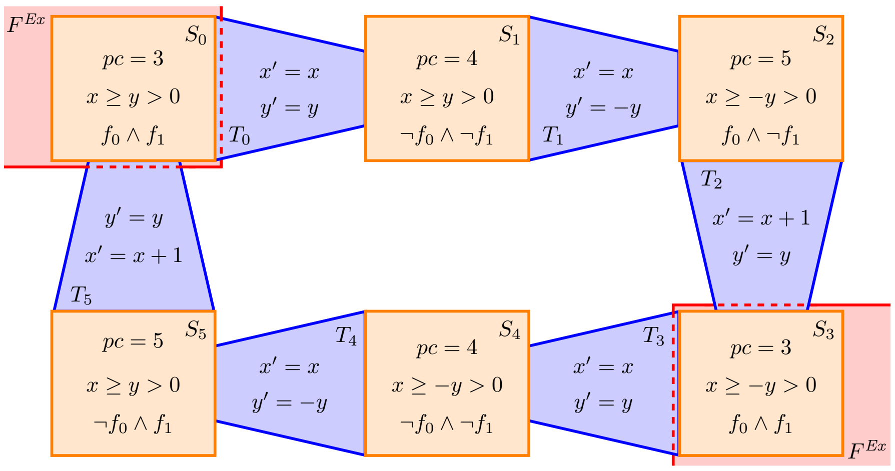

Figure 4: funnel-loop $floop$ of length 6.

0. The first funnel fnl_0 represents the step from location 3 to location 4 of Fig. 2. In S_0 both f_0 and f_1 are true, hence S_0 contains only fair states and also $D_5 \doteq S_0$ does. Notice that $x \geq y \wedge y > 0$ implies $x^2 \geq xy$. Therefore, the condition of the while loop is satisfied.

$$S_0 \doteq pc = 3 \wedge x \geq y \wedge y > 0 \wedge f_0 \wedge f_1$$
$$T_0 \doteq pc' = 4 \wedge x' = x \wedge y' = y \wedge \neg f_0' \wedge \neg f_1'$$

1. The second funnel fnl_1 performs the step from $pc = 4$ to $pc = 5$. In this step, the program of Fig. 2 assigns a nondeterministic value to y. The funnel underapproximates this transition by always assigning to y the opposite of its current value. In addition, since $y > 0$ in S_1, the transition relation assigns f_0' to true.

$$S_1 \doteq pc = 4 \wedge x \geq y \wedge y > 0 \wedge \neg f_0 \wedge \neg f_1$$
$$T_1 \doteq pc' = 5 \wedge x' = x \wedge y' = -y \wedge f_0' \wedge \neg f_1'$$

2. The third funnel fnl_2 performs the last step of the first iteration of the while loop. Its transition relation increases the value of x by one and,

since $y < 0$ holds in the current state, f_1 is true in the next one.

$$S_2 \doteq pc = 5 \wedge x \geq -y \wedge y < 0 \wedge f_0 \wedge \neg f_1$$
$$T_2 \doteq pc' = 3 \wedge x' = x + 1 \wedge y' = y \wedge f_0' \wedge f_1'$$

3. The fourth funnel fnl_3 represents the first step of the loop of Fig. 2 as fnl_0. However, in this case y is negative.

$$S_3 \doteq pc = 3 \wedge x \geq -y \wedge y < 0 \wedge f_0 \wedge f_1$$
$$T_3 \doteq pc' = 4 \wedge x' = x \wedge y' = y \wedge \neg f_0' \wedge \neg f_1'$$

4. The fifth funnel fnl_4 is analogous to fnl_1, but has negative value of y.

$$S_4 \doteq pc = 4 \wedge x \geq -y \wedge y < 0 \wedge \neg f_0 \wedge \neg f_1$$
$$T_4 \doteq pc' = 5 \wedge x' = x \wedge y' = -y \wedge \neg f_0' \wedge f_1'$$

5. Finally, funnel fnl_5 is analogous to fnl_2, but has positive value of y.

$$S_5 \doteq pc = 5 \wedge x \geq y \wedge y > 0 \wedge \neg f_0 \wedge f_1$$
$$T_5 \doteq pc' = 3 \wedge x' = x + 1 \wedge y' = y \wedge f_0' \wedge f_1'$$

It can be easily observed that each funnel satisfies all hypotheses of Def. 1 and the funnels are correctly chained (Def. 2) by definition of the destination regions. Notice that every region and transition of *floop* is a purely conjunctive formula and both S_0 and S_3 underapproximate the fair states. Therefore, in every iteration through *floop* we visit the fair states twice, in S_0 with positive y and in S_3 with negative y. *floop* satisfies all hypotheses of Th. 1 and represents at least one counterexample for our initial LTL model checking problem.

It is possible to define a funnel-loop composed of a single funnel $fnl \doteq \langle S, T, D, \mathrm{R}\textsc{f} \rangle$, where the components can be defined in terms of the funnels we defined above as follows. The source region is the union of the source regions of the $\{fnl_i\}_{i=0}^{5}$: $S \doteq \bigvee_{i=0}^{5} S_i$. The destination region is the last destination region of *floop*: $D \doteq D_5$. The transition relation can be defined as $T \doteq \bigvee_{i=0}^{5}(S_i \wedge T_i)$

by observing that the source regions $\{S_i\}_{i=0}^5$ are pairwise-disjoint. Finally, the ranking function $\textsc{Rf}$ is defined as a function that maps every assignment to the symbols in V to a number in $\mathbb{N}$ such that it assigns decreasing values to states in the regions $S_0, \ldots, S_4$ and assigns the constant 0 to states in S_5:

$$\textsc{Rf}(V) \doteq \begin{cases} 0 & \text{if } S_5(V), \\ 1 & \text{if } S_4(V), \\ 2 & \text{if } S_3(V), \\ 3 & \text{if } S_2(V), \\ 4 & \text{if } S_1(V), \\ 5 & \text{otherwise.} \end{cases}$$

By construction the transition relation maps every state in S_0 to some state in S_1, which is in turn mapped into S_2 and so on. Therefore, every state in $S \wedge \textsc{Rf} > 0$ is mapped to some other state in S in which the ranking function has lower value. $S \wedge \textsc{Rf} = 0$ is equivalent to S_5 and in such region T corresponds to T_5. Therefore, in a single transition we reach D_5 that, by definition, is equivalent to D and contained in S_0.

6. Model decomposition via Existential Components

In the previous section we segmented the paths of a fair transition system into funnels representing finite paths. In the following we adopt an orthogonal view and decompose the system with respect to a partitioning of its symbols. For each set of symbols, an *existential component* (E-component) describes their behaviour with respect to a set of regions and represents a set of loops over such regions. Each E-component represents some infinite behaviour that a subset of the symbols can exhibit, provided that all other symbols satisfy a set of assumptions. Therefore, while funnels describe sets of finite paths, E-components describe (possibly empty) sets of infinite paths.

We will show how E-components can be obtained from funnel-loops with an additional restriction on their transition relation, hence how an E-component can be constructed by concatenating funnels.

We then compose E-components to obtain another E-component whose loops consider the union of the symbols of the smaller ones. We compose them until we obtain a component considering all the symbols of the system. Among all its loops we search for one that is also fair. We then restrict its language to only fair paths by projecting the E-component over the regions of the fair loop. We show that such E-component corresponds to a funnel-loop for the transition system, hence proving that it admits at least one fair path.

We first define the structure and properties of the E-components and we show under which conditions a funnel-loop corresponds to an E-component. Then, we define the composition and projection operators for E-components and, finally, we show how such operators allow the representation of a funnel-loop that satisfies all hypotheses of Th. 1.

6.1. E-component

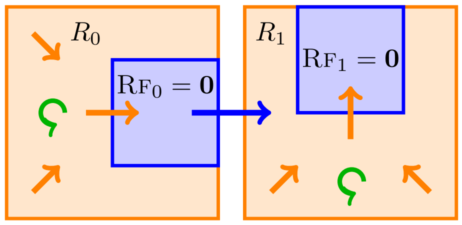

Figure 5: E-component with two regions showing the three kinds of transitions.

An E-component is a transition system associated with a set of regions, assumptions and ranking functions. We call the conjunction of a region and its corresponding assumption *restricted region* and, in addition, E-components associate to each restricted region a ranking function. An E-component is such that its restricted regions group states with "similar behaviour" with respect to the transition relation. If some state in a restricted region allows for a transition with certain characteristics, then a transition with the same characteristics must exist for all states in the restricted region, hence the name existential components. In the following, we first describe in more detail what we mean by similar behaviour via the definition of three predicates that classify transitions. Then, we employ these predicates to formally define E-components. Finally, we characterise the language of such components.

We are interested in transitions representing self-loops over the restricted regions of two types: self-loops in which the ranking function decreases and self-loops in which the ranking function remains constant. We call them *ranked* and *stutter* transitions respectively and characterise them using two relations $\mathit{rankedT}_j(V, V')$ and $\mathit{stutterT}_j(V, V')$ over symbols V and V'. A transition in the restricted region with index j is a ranked transition iff $\mathit{rankedT}_j$ holds and it is a stutter transition iff $\mathit{stutterT}_j$ does. Finally, we consider transitions between possibly distinct restricted regions, starting from a state in which the ranking function is $\mathbf{0}$ and reaching some state in the second region. We call them *progress* transitions and characterise them using the predicate $\mathit{progressT}_{j,j'}(V, V')$. We call a transition a progress transition from region j to region j' iff $\mathit{progressT}_{j,j'}$ holds. Therefore, we distinguish three kinds of transitions between regions and require that either no state allows for a transition of a given kind or all states in the same restricted region admit such a transition. Fig. 5 depicts an E-component with two regions and the three different kinds of transitions.

We now proceed to formally define E-components and their operators. For a set of symbols V, let $\mathcal{R} \doteq \{R_j(V)\}_{j=0}^{m-1}$ be the set of regions, $\mathcal{A} \doteq \{A_j(V)\}_{j=0}^{m-1}$ be the set of assumptions and $\mathcal{W} \doteq \{\mathrm{RF}_j(V)\}_{j=0}^{m-1}$ be the set of ranking functions. Then, $R_j \wedge A_j$ is the j^{th} restricted region and RF_j is the ranking function associated to it. We define the three relations that classify the transitions as follows:

$$\begin{aligned}
\mathit{rankedT}_j(V, V') &\doteq R_j \wedge A_j \wedge \mathbf{0}_j <_j \mathrm{RF}_j \wedge R'_j \wedge A'_j \wedge \mathrm{RF}'_j < \mathrm{RF}_j \\
\mathit{stutterT}_j(V, V') &\doteq R_j \wedge A_j \wedge R'_j \wedge A'_j \wedge \mathrm{RF}'_j = \mathrm{RF}_j \\
\mathit{progressT}_{j,j'}(V, V') &\doteq R_j \wedge A_j \wedge \mathbf{0}_j = \mathrm{RF}_j \wedge R'_{j'} \wedge A'_{j'}
\end{aligned}$$

Notice that the relations $\mathit{rankedT}_j$ and $\mathit{sutterT}_j$ are always disjoint; in the first case the ranking function strictly decreases, while in the second one it must remain constant. However, they are not a partitioning of all possible transitions. In fact, transitions in which the ranking function increases or that move to another region are in neither of the two sets of transitions. In addition, $\mathit{progressT}_{j,j'}$ and $\mathit{rankedT}_j$ are always disjoint by definition, while the first one could have a non-empty intersection with $\mathit{sutterT}_j$ if $j = j'$. In particular, all transitions that both start and end in a state satisfying $R_j \wedge A_j \wedge \mathrm{RF}_j = \mathbf{0}_j$ are in the intersection of $\mathit{stutterT}_j$ and $\mathit{progressT}_{j,j}$. Therefore, the existence of one such transition implies that all states in the restricted region must allow for at least one stutter transition. In addition, for the states in which $\mathrm{RF}_j = \mathbf{0}_j$, this transition is also a progress transition, hence they all admit at least one progress transition that remains in the same region.

We remark that E-components represent the possibility of performing such transitions: they group states for which there exists a successor along the same transition types.

Given a partitioning $\{V^i\}_{i=0}^n$ of the symbols V we want to define the restricted regions such that they allow a set of next assignments to the symbols in a single partition V^i, while the assignment to the symbols in $V^{\neq i} \doteq V \setminus V^i$ is abstracted and only the assumptions are retained. For this reason, we introduce a quantifier alternation ($\exists V^{i'} \forall V^{\neq i'}$), and require the existence of a transition of the given type for every assignment to the $V^{\neq i'}$ satisfying the corresponding assumptions. Therefore, we now formally define E-components as follows.

Definition 3. *E-component. Given a set of symbols V such that $\{V^i\}_{i=0}^n$ is a partitioning of V for some $n \in \mathbb{N}$. An E-component H^i of length $m^i \in \mathbb{N}$ and responsible for V^i is a transition system $\langle V, I^i(V), T^i(V, V')\rangle$ associated with:*

- *a set of regions $\mathcal{R}^i \doteq \{R^i_j(V) \mid 0 \leq j < m^i\}$,*
- *a set of assumptions $\mathcal{A}^i \doteq \{A^i_j(V^{\neq i}) \mid 0 \leq j < m^i\}$, where $V^{\neq i} \doteq \bigcup_{0 \leq k < n, k \neq i} V^k$ and $A^i_j(V^{\neq i}) \doteq \bigwedge_{0 \leq k < n, k \neq i} A^{i,k}_j(V^k)$*
- *a set of functions $\mathcal{W}^i \doteq \{\mathrm{RF}^i_j(V) \mid 0 \leq j < m^i\}$ such that each RF^i_j is a ranking function with respect to a well-founded relation $<^i_j$ and minimal element $\mathbf{0}^i_j$.*

such that the following hold:

I *. The set of initial states $I^i(V)$ of H^i is a subset of the union of the restricted regions:*

$$H^i \models \bigvee_{j=0}^{m^i-1} R^i_j \wedge A^i_j$$

II *. Either no state admits a ranked transition or all states do.*

$$\begin{aligned}
&\forall j : 0 \leq j < m^i \rightarrow \\
&\quad \exists V, V' : \mathit{rankedT}_j(V, V') \quad \models \\
&\quad \forall V \exists V^{i\prime} \forall V^{\neq i\prime} : R^i_j \wedge A^i_j \wedge \mathbf{0}^i_j <^i_j \mathrm{RF}^i_j \wedge A^{i\prime}_j \rightarrow R^{i\prime}_j \wedge T^i \wedge \mathrm{RF}^{i\prime}_j <^i_j \mathrm{RF}^i_j
\end{aligned}$$

III *. Either no state admits a stutter transition or all states do.*

$$\begin{aligned}
&\forall j : 0 \leq j < m^i \rightarrow \\
&\quad \exists V, V' : \mathit{stutterT}_j(V, V') \quad \models \\
&\quad \forall V \exists V^{i\prime} \forall V^{\neq i\prime} : R^i_j \wedge A^i_j \wedge A^{i\prime}_j \rightarrow R^{i\prime}_j \wedge T^i \wedge \mathrm{RF}^{i\prime}_j = \mathrm{RF}^i_j
\end{aligned}$$

IV *. All states admit progress transitions with the same destination regions: they reach the same restricted regions.*

$$\begin{aligned}
&\forall j, j' : 0 \leq j < m^i \wedge 0 \leq j' < m^i \rightarrow \\
&\quad \exists V, V' : \mathit{progressT}_{j,j'}(V, V') \quad \models \\
&\quad \forall V \exists V^{i\prime} \forall V^{\neq i\prime} : R^i_j \wedge A^i_j \wedge \mathrm{RF}^i_j = \mathbf{0}^i_j \wedge A^{i\prime}_{j'} \rightarrow R^{i\prime}_{j'} \wedge T^i
\end{aligned}$$

When clear from the context we will simply write $\mathbf{0}$ and $<$ for $\mathbf{0}^i_j$ and $<^i_j$ respectively. In the definition, each assumption $A^i_j(V^{\neq i})$ of E-component i at index j is composed of n conjuncts $\{A^{i,k}_j(V^k)\}_{0 \leq k < n, k \neq i}$ where each conjunct is a formula over the symbols in a single partition V^k different from V^i.

We define the language of an E-component $H \doteq \langle V, I, T \rangle$ over $\mathcal{R}$, $\mathcal{A}$ and $\mathcal{W}$, written $\mathcal{L}(H)$, as the language of the corresponding transition system $M \doteq \langle V, I, T^M, \top \rangle$, where T^M is defined as follows:

$$T^M \doteq T \wedge (\bigvee_{j=0}^{m-1} R'_j \wedge A'_j) \wedge \bigwedge_{j=0}^{m-1} (R_j \wedge A_j \wedge \mathbf{0} < \mathrm{RF}_j) \rightarrow (R'_j \wedge A'_j \wedge \mathrm{RF}'_j \leq \mathrm{RF}_j)$$

Therefore, we consider only paths that remain within the set of restricted regions and move from one region to another only if the corresponding ranking function is equal to the minimal element: we can perform only ranked or stutter transitions as long as the ranking function corresponding to the current region is greater than its minimal element.

As for funnel-loops, also the definition of E-component allows for regions with non-empty intersection. Similarly to the previous case, this eases the construction of these structures since it has more permissive constraints. However, for every E-component there exists one with pairwise-disjoint regions that admits the same language.[6] For this reason, when proving statements about the language of these structures, we assume without loss of generality the regions of the E-components to be pairwise-disjoint.

6.2. Example decomposition

We now describe a possible decomposition of our running example (Sec. 4) into E-components. The fair transition system Ex is defined over the set of variables $V \doteq \{x, y, pc, f_0, f_1\}$. We consider one variable at a time and define a component representing some of its possible behaviours in the system. It is possible to define many different components for every subset of the symbols, for the sake of brevity and clarity we only describe one for each symbol. In the following E-components we implicitly define every set of initial states as the disjunction of the regions and every ranking function as always equal to its minimal element, hence the E-components will admit no ranked transition.

Consider first the program counter pc. From the transition relation of Ex it is immediately apparent that the variable will keep assuming the values $[3, 4, 5]$ in this order. For this reason we define a E-component H^{pc}, depicted in Fig. 6.

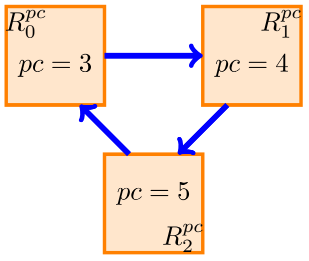

Figure 6: E-component responsible for pc.

[6]See Appendix B.2 for a proof.

H^{pc} is responsible for pc and its three regions are defined as $R_0^{pc} \doteq pc = 3$, $R_1^{pc} \doteq pc = 4$ and $R_2^{pc} \doteq pc = 5$. Then, its transition relation is the disjunction of the 3 progress transitions between the regions: $T^{pc} \doteq (pc = 3 \wedge pc' = 4) \vee (pc = 4 \wedge pc' = 5) \vee (pc = 5 \wedge pc' = 3)$. We do not introduce any self-loop on the regions, since none exists in the transition relation of Ex. Finally, this behaviour does not require any assumption. In fact, the transition relation T^{pc} is sufficient to ensure that we move from one region to another without having to assume anything about the other symbols.

Consider now the Boolean symbols f_0 and f_1. In this case, we define two E-components: H^{f_0} for f_0 and H^{f_1} for f_1. The two E-components are shown in Fig. 7. In both E-components we need to distinguish the truth value of the two symbols in order to identify the fair states, hence we define each E-component using two regions. For $i \in \{0, 1\}$, let $R_0^{f_i}$, $R_1^{f_i}$ be the regions of H^{f_i} and T^{f_i} its transition relation. We define the two regions such that one corresponds to the case in which the variable is assigned to true and the other to the case in which the variable is false. In Ex the two variables can remain constant for any number of steps and toggle their truth value when a certain condition is met. The simplest components we can define in this case are defined as $R_0^{f_i} \doteq f_i$, $R_1^{f_i} \doteq \neg f_i$ and $T^{f_i} \doteq \top$, for $i \in \{0, 1\}$, with no assumptions on the other symbols.

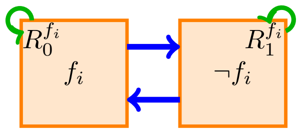

Figure 7: E-components responsible for f_i.

Consider now the variable y and we define H^y as the E-component responsible for such variable. In the transition relation of Ex the variable appears in the following predicates $\{y < 0, y > 0, x^2 \geq xy, y' = y\}$. In only one case it appears together with another symbol: $x^2 \geq xy$. We can observe that if $|x| \geq |y|$ then the predicate must hold. This suggests a dependency between x and y and for this reason we could define a single E-component that considers both symbols together. However, we would like to keep them separated for this example. We break the dependency between the two symbols by considering the stronger conditions $x \geq 1$ and $y \leq 1$. Then, the presence of $y < 0$ and $y > 0$ suggests the need for two regions to distinguish the sign of the variable. Fig. 8 depicts H^y. The E-component has two regions: $R_0^y \doteq y = -1$ and $R_1^y \doteq y = 1$. The regions differentiate the two cases and we introduce two corresponding assumptions $A_0^y \doteq x \geq 1$ and $A_1^y \doteq x \geq 1$. Finally, we define the transition relation T^y of H^y such that it allows stutter transitions in both regions and also progress transitions to move from one region to the other: $T^y \doteq y' = y \vee y' = -y$.

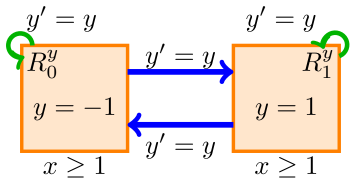

Figure 8: E-component responsible for y.

The only remaining symbol is x, for which we define the E-component H^x depicted in Fig 9. In the transition relation of Ex the variable appears in the following predicates $\{x^2 \geq xy, x' = x, x' = x + 1\}$. We apply the same reasoning as above to analyse the predicate $x^2 \geq xy$ and obtain a single region $R_0^x \doteq x \geq 1$ with assumption $A_0^x \doteq y \leq 1$ for H^x. We define the transition relation T^x of H^x as the disjunction of the two remaining predicates, $T^x \doteq x' = x \vee x' = x + 1$.

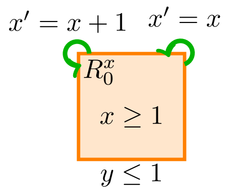

Figure 9: E-component responsible for x.

The purpose of E-components is to split the process of identifying some fair path into two phases. In the first phase, one symbol or one group of closely related symbols should be considered at a time to identify possible infinite behaviours over them, as exemplified above. The successive step requires to identify how they should be composed in order to obtain a structure that represents fair paths of the transition system. For this reason, in §6.4 we introduce two operators over E-components. The operators need to ensure that the components to be combined are compatible and preserve the existence of the infinite behaviours. We achieve this by combining E-components such that the respective assumptions are met. §6.5 shows how the E-components we defined above can be composed to prove the existence of a fair path in Ex.

6.3. From funnel-loops to E-components

The following theorem shows the correspondence between a funnel-loop and an E-component. Therefore, it enables the use of funnels and funnel-loops in the decomposition of a system.

Theorem 3. *Given a set of symbols $\hat{V} \subseteq V$, a funnel-loop $floop$ composed of funnels $[fnl_i]_{i=0}^{n-1}$ such that all its transition relations are of the form $T_i(V, \hat{V}')$ corresponds to an E-component $H \doteq \langle V, \bigvee_{i=0}^{n-1} S_i, \bigvee_{i=0}^{n-1} S_i \wedge T_i \rangle$ responsible for symbols $\hat{V}$ and associated with regions $\{S_i\}_{i=0}^{n-1}$, ranking functions $\{\mathrm{RF}_i\}_{i=0}^{n-1}$ and assumptions $\{\top\}_{i=0}^{n-1}$.*

Proof. We show that H satisfies all hypotheses of Def. 3.

I By definition all assumptions are $\top$ and the initial states are defined as the union of the regions. Therefore, Hyp. **I** holds.

II Hyp. F.1 ensures that in every region S_i, T_i always allows for a successor state. Therefore, also $\bigvee_{i=0}^{n-1} S_i \wedge T_i$ is left-total in the union of the regions. Hyp. F.3 ensures that every self-loop on S_i decreases the associated ranking function RF_i. If a self-loop exist such transition is a ranked transition and all such transitions are ranked. All such states admit a successor and the successor must decrease the value of the ranking function. Therefore, Hyp. **II** holds.

III As observed in the previous case, all self-loops on a region must decrease the corresponding transition relation. Therefore, H admits no stutter transitions and Hyp. **III** holds.

IV Hyp. F.4 ensures that from every region S_i when the ranking function RF_i is equal to $\mathbf{0}$, in one transition T_i we always reach a state in D_i and, by hypotheses FL.1 and FL.2, such state is in the following region $S_{(i+1)\%n}$. Since, the transition relation is left-total by Hyp. F.1, then all states in $S_i \wedge \mathrm{RF}_i = \mathbf{0}$ admit at least one and only successors in $S_{(i+1)\%n}$. Therefore, Hyp. **IV** holds.

□

6.4. Operators over E-component

We now define the projection and composition operators for E-components. Intuitively, the first operator shrinks an E-component by considering only a subset of its regions, while the second operator computes the product of n E-components. These two operators will be useful to identify an E-component that meets some additional requirements in order to represent a funnel-loop.

E-component projection. We define a projection operation for E-components that can be used to obtain a smaller E-component describing a subset of the paths of the original structure. We project an E-component over an ordered subset of its regions, then we restrict the transition relation by removing all stuttering transitions and such that the progress transitions must follow the ordering of the regions and from the last region they can only reach the first one. Therefore, the projection restricts the language of an E-component to the paths that visit only regions in the sequence in order and are either finite or reach the last region infinitely often.

Definition 4. *Given an E-component $H \doteq \langle V, I, T \rangle$ over m regions $\mathcal{R}$, assumptions $\mathcal{A}$ and ranking functions $\mathcal{W}$, we define its projection to a sequence of k indexes $idxs \doteq \langle j^{\downarrow}_0, \ldots, j^{\downarrow}_{k-1} \rangle$ such that $idxs \subseteq \{0, \ldots, m-1\}$ as the E-component $H^{\downarrow} \doteq \langle V, I^{\downarrow}, T^{\downarrow} \rangle$ associated with regions $\mathcal{R}^{\downarrow}$, assumptions $\mathcal{A}^{\downarrow}$ and ranking functions $\mathcal{W}^{\downarrow}$ defined as follows:*

- $I^{\downarrow} \doteq I \wedge \bigvee_{j \in idxs} (R_j \wedge A_j)$;
- $T^{\downarrow} \doteq T \wedge \bigwedge_{h=0}^{k-1} R_{j^{\downarrow}_h} \rightarrow ((R'_{j^{\downarrow}_h} \wedge \mathrm{RF}'_{j^{\downarrow}_h} < \mathrm{RF}_{j^{\downarrow}_h}) \vee (\mathrm{RF}_{j^{\downarrow}_h} = \mathbf{0} \wedge R'_{j^{\downarrow}_{(h+1)\%k}}))$
- $\mathcal{R}^{\downarrow} \doteq \{R_j \mid j \in idxs \wedge R_j \in \mathcal{R}\}$;
- $\mathcal{A}^{\downarrow} \doteq \{A_j \mid j \in idxs \wedge A_j \in \mathcal{A}\}$;
- $\mathcal{W}^{\downarrow} \doteq \{\mathrm{RF}_j \mid j \in idxs \wedge \mathrm{RF}_j \in \mathcal{W}\}$.

Notice that in the projection we restrict the set of initial states to only those in one of the restricted regions corresponding to the indexes $idxs$, and the transition relation is strengthened such that it imposes that the regions in *idxs* are always visited in order. In addition, the projection operator does not modify the regions, assumptions and ranking function of an E-component, but considers a subset of them.

Theorem 4. *The projection $H^{\downarrow}$ over indexes idxs of an E-component H over regions $\mathcal{R}$, assumptions $\mathcal{A}$ and ranking functions $\mathcal{W}$ is an E-component.*

The proof of Th. 4 is reported in Appendix B.3.

E-component composition. We compose E-components such that they meet their respective assumptions. Given a set $\{H^i\}_{i=0}^n$ of E-components, we say that a set of transitions from regions $\{R^i_{j_i}\}_{i=0}^n$ to regions $\{R^i_{j'_i}\}_{i=0}^n$ are *compatible*, if every transition T^i ensures that $\bigwedge_{s=0,s\neq i}^n A^{s,i}_{j'_s}$ holds. In addition, we compose restricted regions of E-components iff the corresponding ranking functions are independent: it is possible to decrease one independently from the others. In the following we define two binary predicates $compatible_{\{H^i\}_{i=0}^n}$ and $indepRank_{\{H^i\}_{i=0}^n}$ that hold iff the two conditions are met.

Definition 5 (compatible transitions). *Let $\{H^i\}_{i=0}^n$ be a set of E-components such that $\{V^i\}_{i=0}^n$ are pairwise disjoint and $\bigcup_{i=0}^n V^i \subseteq V$. A transition from state $\boldsymbol{v}$ to $\boldsymbol{v}'$ is* compatible *iff the transitions of the E-components, from every pair of states in the same regions, meet the respective* *assumptions of the E-components.*

$$\mathit{compatible}_{\{H^i\}_{i=0}^n}(\hat{V},\hat{V}')\doteq\forall V,V' : \underbrace{\bigwedge_{0\leq j_0<m^0,0\leq j_0'<m^0,\ldots,0\leq j_n<m^n,0\leq j_n'<m^n}}_{\textit{all possible pair of indexes for the E-components } \{H^i\}_{i=0}^n}$$

$$(\bigwedge_{i=0}^n \underbrace{R^i_{j_i}(\hat{V}) \wedge A^i_{j_i}(\hat{V}^{\neq i}) \wedge R^i_{j_i'}(\hat{V}') \wedge A^i_{j_i'}(\hat{V}^{\neq i\prime})}_{j_i, j_i' \textit{ containing both } \hat{V} \textit{ and } \hat{V}'} \wedge$$

$$\underbrace{R^i_{j_i}(V) \wedge A^i_{j_i}(V^{\neq i}) \wedge R^i_{j_i'}(V') \wedge A^i_{j_i'}({V^{\neq\{h\}_{h=0}^n}}') \wedge T^i(V,V')}_{\substack{\textit{for all } V \textit{ in } j_i,\ V' \textit{ in } j_i' \textit{ such that } V,V' \models T^i \textit{ and } V' \textit{ meets all} \\ \textit{assumptions of } H^i \textit{ at } j_i' \textit{ on symbols of E-components not in } \{H^i\}_{i=0}^n}} \quad \wedge$$

$$\underbrace{(\mathrm{RF}^i_{j_i'}(\hat{V}') < \mathrm{RF}^i_{j_i}(\hat{V}) \leftrightarrow \mathrm{RF}^i_{j_i'}(V') < \mathrm{RF}^i_{j_i}(V))}_{\textit{transition } V,V' \textit{ of the same type of transition} \hat{V},\hat{V}'} \wedge$$

$$(\mathbf{0} < \mathrm{RF}^i_{j_i}(\hat{V}) \leftrightarrow \mathbf{0} < \mathrm{RF}^i_{j_i}(V)) \wedge (\mathbf{0} < \mathrm{RF}^i_{j_i'}(\hat{V}') \leftrightarrow \mathbf{0} < \mathrm{RF}^i_{j_i'}(V'))) \rightarrow$$

$$\bigwedge_{i=0}^n \qquad \underbrace{\bigwedge_{h=0,h\neq i}^n A^{i,h}_{j_i'}(V^{h\prime})}_{\textit{all assumptions of } H^i \textit{ on the } \{V^h\}_{h=0}^n \textit{ are met}}.$$

A set of transitions has *independent ranks* if it is possible to decrease each ranking function independently from the others. Consider the restricted regions $\{R^i_{j_i} \wedge A^i_{j_i}\}_{i=0}^n$, there exist transitions with *independent ranks* if, for each $\mathrm{RF}^{i_r}_{j_{i_r}}$ with $0 \leq i_r \leq n$, it is possible to perform a self-loop on the conjunction of the restricted regions $\bigwedge_{i=0}^n R^i_{j_i} \wedge A^i_{j_i}$ such that $\mathrm{RF}^{i_r}_{j_{i_r}}$ decreases and all the other ranking functions remain constant: $\bigwedge_{i=0,i\neq i_r}^n {\mathrm{RF}^i_{j_i}}' = \mathrm{RF}^i_{j_i}$.

Definition 6 (independent ranks). *Let $\{H^i\}_{i=0}^n$ be a set of the E-components such that $\{V^i\}_{i=0}^n$ are pairwise disjoint and $\bigcup_{i=0}^n V^i \subseteq V$. A self-loop over the intersection of the restricted regions has* independent ranks *iff for every ranking function there exists a compatible conjunction of the* *transitions decreasing only that function.*

$$\mathit{indepRank}_{\{H^i\}_{i=0}^n}(\hat{V},\hat{V}') \doteq \underbrace{\bigwedge_{0\le j_0<m^0,\ldots,0\le j_n<m^n}}_{\text{all possible indexes for the E-components } \{H^i\}_{i=0}^n}$$

$$((\quad \underbrace{(\sum_{i=0}^{n} \mathrm{RF}^i_{j_i})(\hat{V}') < (\sum_{i=0}^{n} \mathrm{RF}^i_{j_i})(\hat{V})}_{\text{some ranking function decreases, all others remain constant}} \quad \wedge$$

$$\underbrace{\bigwedge_{i=0}^{n} R^i_{j_i}(\hat{V}) \wedge A^i_{j_i}(\hat{V}^{\neq i}) \wedge R^i_{j_i}(\hat{V}') \wedge A^i_{j_i}(\hat{V}^{\neq i\prime}))}_{\hat{V},\hat{V}' \text{are in restricted regions } j_i,j_i'} \rightarrow$$

$$\bigwedge_{i=0}^{n} \underbrace{(\forall V : (\bigwedge_{h=0}^{n} R^h_{j_h}(V) \wedge A^h_{j_h}(V^{\neq h})) \rightarrow \mathrm{RF}^i_{j_i}(V) = \mathbf{0})}_{\text{current ranking function } \mathrm{RF}^i_{j_i} \text{ is always } \mathbf{0}} \ \vee$$

$$\exists V,V' : \underbrace{(\bigwedge_{h=0}^{n} R^h_{j_h}(V) \wedge A^h_{j_h}(V^{\neq h}) \wedge T^h(V,V') \wedge R^h_{j_h}(V') \wedge A^h_{j_h}(V^{\neq h\prime})) \wedge}_{V,V' \text{ in same restricted regions of } \hat{V},\hat{V}'}$$

$$\underbrace{\mathrm{RF}^i_{j_i}(V') < \mathrm{RF}^i_{j_i}(V) \wedge (\bigwedge_{h=0,h\neq i}^{n} \mathrm{RF}^h_{j_h}(V') = \mathrm{RF}^h_{j_h}(V)) \wedge}_{\text{current ranking function decreases, all others remain constant}}$$

$$\mathit{compatible}_{\{H^k\}_{k=0}^n}(V,V'))$$

The composition operator for a set of E-components $\{H^i\}_{i=0}^n$ requires the corresponding sets $\{V^i\}_{i=0}^n$ to be pairwise disjoint. We write $\{V^i\}_{i\notin\{0,\ldots,n\}}$ for
755 the possibly empty list of other sets to complete the partitioning: $\{V^i\}_{i=0}^n \cup \{V^i\}_{i\notin\{0,\ldots,n\}}$ is a partitioning of V.

Definition 7 (composition of *E*-components). *We define the composition of a set of E-components $\{H^i\}_{i=0}^n$, such that the sets of local symbols $\{V^i\}_{i=0}^n$ are pairwise disjoint, as $H^c \doteq \bigotimes_{i=0}^n H^i = \langle V, I^c, T^c\rangle$ where:*

760 • $V^c \doteq \bigcup_{i=0}^n V^i$.

• *The set of regions is the intersection of the regions and assumptions over*

V^c of the E-components.

$$\mathcal{R}^c \doteq \{\bigwedge_{i=0}^{n} R^i_{j_i} \wedge \bigwedge_{h=0, h\neq i}^{n} A^{i,h}_{j_i} \mid \forall i \in \{0,\ldots,n\}, j_i \in \{0,\ldots,m^i-1\} \ : \ R^i_{j_i} \in \mathcal{R}^i \wedge$$
$$\forall h \in \{0,\ldots,n\} \setminus \{i\} \ : \ A^i_{j_i} \in \mathcal{A}^i \wedge A^{i,h}_{j_i} \in A^i_{j_i}\}$$

- *The set of assumptions is given by the conjunction of the assumptions of the $\{H^i\}_{i=0}^n$ over the symbols not in V^c.*

$$\mathcal{A}^c \doteq \{\bigwedge_{i=0}^{n} \bigwedge_{h\notin\{0,\ldots,n\}} A^{i,h}_{j_i} \mid \forall i \in \{0,\ldots,n\}, h \notin \{0,\ldots,n\}, j_i \in \{0,\ldots,m^i-1\} :$$
$$A^i_{j_i} \in \mathcal{A}^i \wedge A^{i,h}_{j_i} \in A^i_{j_i}\}$$

- *The ranking functions for the regions are obtained by considering the sum of the ones corresponding to the regions of the $\{H^i\}_{i=0}^n$.*

$$\mathcal{W}^c \doteq \{\sum_{i=0}^{n} \mathrm{RF}^i_{j_i} \mid \forall i \in \{0,\ldots,n\}, j_i \in \{0,\ldots,m^i-1\} \ : \ \mathrm{RF}^i_{j_i} \in \mathcal{W}^i\}$$

- *$I^c \doteq \bigwedge_{i=0}^n I^i$;*

- *The set of transitions is given by the conjunction of the transition relations of the $\{H^i\}_{i=0}^n$ restricted to the compatible transitions.*

$$T^c \doteq \mathit{compatible}_{\{H^i\}_{i=0}^n} \wedge \mathit{indepRank}_{\{H^i\}_{i=0}^n} \wedge \bigwedge_{i=0}^{n} T^i$$

Theorem 5. *Given a set of E-components $\{H^i\}_{i=0}^n$, their composition $H^c \doteq \bigotimes_{i=0}^n H^i = \langle V, I^c, T^c\rangle$ is an E-component with respect to regions $\mathcal{R}^c$, assumptions $\mathcal{A}^c$ and ranking functions $\mathcal{W}^c$.*

The proof of Th. 5 is reported in Appendix B.4.

We remark that the definition of the composition operator ensures that H^c admits only transitions that satisfy both *compatible* and *indepRank*. In addition, we consider only simple interactions between the ranking functions of different E-components. It is possible to extend the operator to allow for more complex combinations such as nesting of ranking functions or allowing the ranking function of an E-component to decrease once every time all the other E-components perform a loop over their regions. However, including this kind of compositions would make the definitions and proofs much more complex and with many more cases to be considered.

6.5. Example E-components composition

We now show how the E-components we defined in Subsec. 6.2 can be combined to conclude the existence of a fair path in the fair transition system Ex defined in Sec. 4.

We first compute a E-component H^{f_0,f_1} as $H^{f_0} \otimes H^{f_1}$. H^{f_0} and H^{f_1} have no assumptions and all ranking functions are always equal to their minimal element. Therefore, all transitions are compatible and the result of the composition is the synchronous product of the two E-components. Fig. 10 depicts the resulting E-component H^{f_0,f_1}. H^{f_0,f_1} has four regions, one for each of the possible truth assignments of the two Boolean symbols f_0 and f_1 and it allows all 16 possible transitions and self-loops over them.

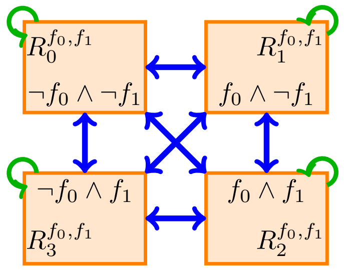

Figure 10: E-component responsible for $\{f_0, f_1\}$.

We now compute $H^{x,y}$ responsible for x and y as the composition of H^x and H^y; the E-component is depicted in Fig. 11. The assumption of H^x requires $y \leq 1$ and both assumptions of H^y require $x \geq 1$. Therefore, it can be easily seen that H^x will always meet the assumptions required by H^y and vice-versa also H^y meets the assumption of H^x. Since the two E-components do not have any assumptions on the other symbols, the resulting E-component $H^{x,y}$ has no assumptions. $H^{x,y}$ has two regions, obtained by the conjunction of the two regions of H^y with the only region of H^x. The transition relation of $H^{x,y}$ is given by the conjunction of the transition relations of H^x and H^y. Both regions of the

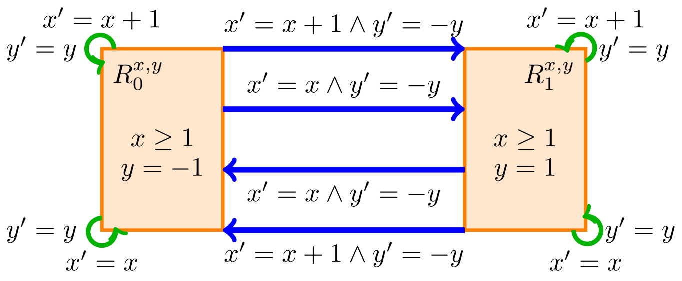

Figure 11: E-component responsible for $\{x, y\}$.

E-component admit stutter transitions of two kinds: one in which both variables x and y remain constant, and one in which y is constant and x increases by one. Notice that the ranking functions of the regions are always constant and equal to the minimal element. Therefore, the transitions satisfy *indepRank* because its definition (Def. 6) is an implication in which the left-hand-side requires at least one ranking function to decrease. Finally, $H^{x,y}$ also admits progress transitions from one region to the other of two kinds: in both cases the value of y changes its sign, while in one case x remains constant and in the other it increments by one.

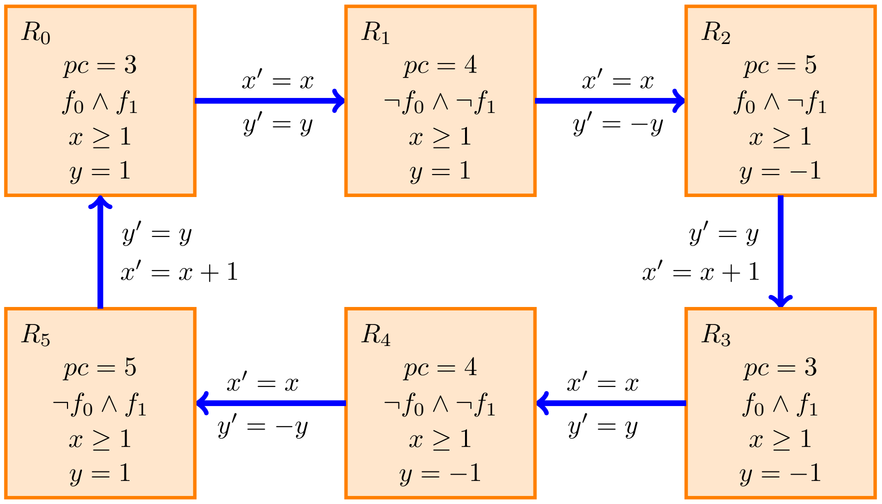

Figure 12: E-component responsible for all symbols.

Finally, we compute $H \doteq H^{pc} \otimes H^{x,y} \otimes H^{f_0,f_1}$. None of the E-components has assumptions and all their ranking functions are always equal to the minimal element. For this reason, all transitions are compatible and have independent ranks. Therefore, the transition relation of H is the conjunction of the transition relations of the 3 E-components. H has 24 regions, given by the product of the 3 regions of H^{pc}, 2 of $H^{x,y}$ and 4 of H_{f_0,f_1}. The regions represent all the different configurations that can be reached by employing compatible transitions of our E-components H^{pc}, H^{f_0}, H^{f_1}, H^x and H^y. Recall that our objective is to identify fair paths for the fair transition system Ex defined in Sec. 4. Not all transitions of H are also transition of Ex. For example, H admits a transition that increases the value of x from states where $pc = 3$, while this is not possible in Ex. However, using the projection operator we can restrict H by considering a subset of its regions. In particular, we are interested in the sequence of regions that would allow us to obtain a representation of at least one fair path for Ex. We select 6 regions and depict the projection of H over such regions in Fig. 12. We call this projection $H^{\downarrow}$. In particular we consider the following regions:

$$
\begin{aligned}
R_0 &\doteq f_0 \wedge f_1 \wedge y = 1 \wedge x \geq 1 \wedge pc = 3,\\
R_1 &\doteq \neg f_0 \wedge \neg f_1 \wedge y = 1 \wedge x \geq 1 \wedge pc = 4,\\
R_2 &\doteq f_0 \wedge \neg f_1 \wedge y = -1 \wedge x \geq 1 \wedge pc = 5,\\
R_3 &\doteq f_0 \wedge f_1 \wedge y = -1 \wedge x \geq 1 \wedge pc = 3,\\
R_4 &\doteq \neg f_0 \wedge \neg f_1 \wedge y = -1 \wedge x \geq 1 \wedge pc = 4,\\
R_5 &\doteq \neg f_0 \wedge f_1 \wedge y = 1 \wedge x \geq 1 \wedge pc = 5.
\end{aligned}
$$

Notice that the 6 regions underapproximate those that we have already considered in the funnel-loop described in §5.3. In particular, for every $i \in \{0, \dots, 5\}$ R_i underapproximates S_i. In fact, there is correspondence between the paths of $H^{\downarrow}$ and those of the funnel-loop. Therefore, $H^{\downarrow}$ proves the existence of a fair path in the language of the fair transition system Ex and we reached our goal. In the following we formalise this relationship between E-components and funnel-loops. Th. 6 details the conditions under which an E-component implies the existence of a funnel-loop proving the non-emptiness of the language of a fair transition system.

820 *6.6. From E-components to funnel-loops*

We now provide a sequence of sufficient conditions for an E-component $H \doteq \langle V, I, T \rangle$ over regions $\mathcal{R}$, assumptions $\mathcal{A}$ and ranking functions $\mathcal{W}$ to describe a funnel-loop.

Theorem 6. *Let M be a given fair transition system $M \doteq \langle V, I^M, T^M, F^M \rangle$.* 825 *The existence of an E-component $H \doteq \langle V, I, T \rangle$ responsible for all symbols V over regions $\mathcal{R}$ and ranking functions $\mathcal{W}$ of length m satisfying all the following conditions, implies the existence of a funnel-loop for M, hence the existence of at least one fair path in M.*

H.0 *An initial state of H is reachable in M:*

$$M \rightsquigarrow I$$

830 ***H.1*** *The transition relation of H, restricted to the transitions that follow the sequence of regions $\mathcal{R}$, underapproximates the transition relation of M:*

$$\forall j, V, V' : R_j \wedge T \wedge ((\textrm{R}\textsc{f}'_j < \textrm{R}\textsc{f} \wedge R'_j) \vee (\textrm{R}\textsc{f}_j = \mathbf{0} \wedge R'_{(j+1)\%m})) \rightarrow T^M$$

H.2 *From the last region in $\mathcal{R}$, H can only reach fair states.*

$$\forall V, V' : R_{m-1} \wedge \textrm{R}\textsc{f}_{m-1} = \mathbf{0} \wedge T \wedge R'_0 \rightarrow F^{M'}$$

H.3 *There must exist a transition from each R_j with $\textrm{R}\textsc{f}_j = \mathbf{0}$ to the following region $R_{(j+1)\%m}$:*

$$\forall j \exists V, V' : 0 \leq j < n-1 \rightarrow R_j \wedge \textrm{R}\textsc{f}_j = \mathbf{0} \wedge T \wedge R'_{(j+1)\%m}$$

835 ***H.4*** *If a region has a non-trivial ranking function, then it must be possible to decrease it:*

$$\forall j : 0 \leq j < m \rightarrow (\forall V : \textrm{R}\textsc{f}_j = \mathbf{0}) \vee (\exists V, V' : R_j \wedge T \wedge R'_j \wedge \textrm{R}\textsc{f}'_j < \textrm{R}\textsc{f})$$

Proof. Since H is responsible for all symbols V, then all assumptions in $\mathcal{A}$ are empty. We first define the funnel-loop $floop$ corresponding to the E-component H and then prove that: all of its funnels meet the hypotheses of Def. 1, $floop$ is indeed a funnel-loop (Def. 2) and $floop$ meets all the hypotheses of Th. 1.

Define $floop$ as the concatenation of the funnels $\{fnl_j\}_{j=0}^{m-1}$. Each funnel fnl_j is the 4-tuple $\langle S_j, T_j, D_j, \textsc{Rf}_j \rangle$ such that:

- $S_j \doteq R_j$ for $R_j \in \mathcal{R}$;
- $T_j \doteq T \wedge ((\textsc{Rf}'_j < \textsc{Rf}_j \wedge R'_j) \vee (\textsc{Rf}_j = \mathbf{0} \wedge R'_{(j+1)\%m}))$;
- $D_j \doteq \exists V' : R_j \wedge \textsc{Rf}_j = \mathbf{0} \wedge T \wedge R'_{(j+1)\%m}$;
- $\textsc{Rf}_j \in \mathcal{W}$.

We show that each fnl_j is a funnel (Def. 1).

F.1 T_j is left-total with respect to S_j because T always allows for at least one successor that is either in the same region with decreasing ranking function or in the following region. H is an E-component, hence it satisfies Hyp. **II** and Hyp. **IV**. Hypotheses H.4 and H.3 ensure that at least one transition of both kinds exists in H. Therefore, from every state in S_j with $\mathbf{0} < \textsc{Rf}_j$ there exists a successor in the same region with $\textsc{Rf}'_j < \textsc{Rf}_j$ and from every state in S_j with $\textsc{Rf}_j = \mathbf{0}$, T admits a successor in $S_{(j+1)\%m}$.

F.2 holds by construction of T_j; $\mathbf{0} < \textsc{Rf}_j$ implies that the second component of the disjunction in T_j is false and T_j becomes equivalent to $T \wedge \textsc{Rf}'_j < \textsc{Rf}_j \wedge R'_j$ which implies R'_j.

F.3 holds by construction of T_j; $\mathbf{0} < \textsc{Rf}_j$ implies that the second component of the disjunction in T_j is false and T_j becomes equivalent to $T \wedge \textsc{Rf}'_j < \textsc{Rf}_j \wedge R'_j$ which implies $\textsc{Rf}'_j < \textsc{Rf}_j$.

F.4 holds by construction of D_i: we defined it as the existential image of $R_j \wedge \textsc{Rf}_j = \mathbf{0}$ with respect to $T \wedge R'_{(j+1)\%m}$.

We now show that $floop$ is a funnel-loop (Def. 2).

FL.1, FL.2 By construction each T_j, from a state in $R_j \wedge \textsc{Rf}_j = \mathbf{0}$ with $j < m$ can only reach states that are in $R_{(j+1)\%m}$. Therefore, by construction of D_j both Hyp. FL.1 (for $j < m - 1$ and Hyp. FL.2 (for $j = m - 1$) hold.

Finally, we show that $floop$ meets all hypotheses of Th. 1.

FF.1 Hyp. **I** ensures that the initial states of H underapproximate the union of its regions. Hyp. H.0 ensures that there exist a reachable initial state in H. Therefore, there is a reachable state in the union of the regions and Hyp. FF.1 holds.

FF.2 D_{m-1} is defined as the existential image of $R_{m-1} \wedge \textsc{Rf}_{m-1} = \mathbf{0}$ with respect to $T \wedge R'_0$. Hyp. H.2 ensures that all such states are also fair, hence Hyp. FF.2 holds.

FF.3 By construction each $S_j \wedge T_j$ underapproximates T. Hyp. H.1 ensures that T underapproximates T^M. Therefore each $S_j \wedge T_j$ underapproximates T^M.

$\square$

7. Search procedure

This section describes the procedure for the synthesis of a funnel-loop. Given a fair transition system M and a set of E-components $\mathcal{H}$, the procedure tries to find, in a fully automated manner, a funnel-loop *fnl_loop* for M and a finite path of M ending in a region of *fnl_loop*. $\mathcal{H}$ is a possibly empty set of E-components provided by the user to guide the search. For this reason we will refer to them as *hints*. The procedure selects a possibly empty subset of hints and uses them as building blocks to define the funnel-loop while synthesising the missing components. When the set of hints is empty the procedure identifies a funnel-loop for a fair transition system without relying on any additional information. In the following, we call trivial hint the E-component $H \doteq \langle V, \top, \top \rangle$ responsible for no symbols ($V^H \doteq \emptyset$) such that all its regions and assumptions are the constant $\top$ and all its ranking functions are always $\mathbf{0}$.

Algorithm 1 SEARCH-FUNNEL-LOOP(M, $\mathcal{H}$)

```
    ▷ Iterate over candidate loops of increasing length.
 1: for $\langle prefix, loop\_r, loop\_t, H\rangle \in$ GENERATE-CANDIDATE-LOOPS($M, \mathcal{H}$) do
 2:     $v_0 \leftarrow prefix[\mathbf{len}(prefix) - 1]$        ▷ Witness for reachability, Hyp. FF.1.
        ▷ Iterate over funnel-loop templates for current candidate loop.
 3:     for $template \in$ GENERATE-TEMPLATES($v_0, loop\_r, loop\_t, H$) do
 4:         $ef\_constrs \leftarrow template.ef\_constraints()$        ▷ Get ∃∀ problem.
 5:         $\langle found, model\rangle \leftarrow$ SEACH-PARAMETER-ASSIGNMENT($ef\_constrs$)
 6:         if $found == \top$ then        ▷ Replace parameters with assignment.
 7:             $fnl\_loop \leftarrow template.instantiate(model)$
 8:             return $\langle prefix, fnl\_loop\rangle$ ▷ Reachability witness and funnel-loop.
 9:         end if
10:     end for
11: end for
12: return $unknown$
```

Alg. 1 describes the main steps of the procedure. We reduce the synthesis problem to a sequence of SMT queries. In order to reduce the search space, given a E-component H we only look for funnel-loops obtained by deterministic completions of H; we strengthen the transition relation of H by adding deterministic assignments to the symbols for which H is not responsible. More in detail, Alg. 1 enumerates candidate conjunctive fair loops of the fair transition system and compositions of E-components that admit such loop (line 1). If GENERATE-CANDIDATE-LOOPS selects no hints or $\mathcal{H}$ is empty the returned H is the trivial hint. For each candidate loop, the procedure generates a sequence of parameterised funnel-loops, called funnel-loop templates, as a strengthening of the corresponding E-component (line 3). The predicates of a funnel-loop template are over the symbols of the system M and a set of parameters P. The procedure searches an assignment to the parameters such that all the hypotheses of Defs. 1 and 2 and of Th. 1 hold. At line 4 the procedure obtains the $\exists\forall$-quantified problem associated with the funnel-loop template and then, at line 5 tries to solve it. Finally, at line 7, it replaces the parameters with the assignment identified at the previous step, thus obtaining the desired funnel-loop.

The procedure relies on ranking functions to perform two different tasks. Alg. 2 tries to synthesise ranking functions to avoid considering candidate loops for which we know a ranking function exists. The existence of the ranking function proves that the loop must eventually terminate, hence it cannot correspond to an infinite path. Then, ranking function templates are also used as components for the funnels of the funnel-loop template generated by Alg. 3.

Before going into the details of the procedure, we first show its application to our running example. We then describe how we represent and enumerate candidate loops and compositions of E-components for the transition system M. After that, we detail how a funnel-loop template is generated from a candidate loop and E-component. Finally, we report the synthesis problem associated with a funnel-loop template.

7.1. Example funnel-loop search

We first recall the definition of the fair transition system Ex we introduced in Sec. 4. Let $V \doteq \{x, y, pc, f_0, f_1\}$ be a set of symbols such that pc and x are integer variables, y has real type and f_0 and f_1 are two Boolean symbols. Then, the fair transition system is $Ex \doteq \langle V, I, T, F\rangle$, where:

$$\begin{aligned}
I &\doteq pc = 3;\\
F &\doteq f_0 \wedge f_1;\\
T &\doteq (pc = 3 \rightarrow (x^2 \geq xy \wedge pc' = 4 \wedge x' = x \wedge y' = y)) \wedge\\
&\quad (pc = 4 \rightarrow (pc' = 5 \wedge x' = x)) \wedge\\
&\quad (pc = 5 \rightarrow (pc' = 3 \wedge x' = x + 1 \wedge y' = y)) \wedge\\
&\quad ((f_0 \wedge f_1) \rightarrow (\neg f_0' \wedge \neg f_1')) \wedge\\
&\quad (f_0' \rightarrow (f_0 \vee y > 0)) \ \wedge \ (f_1' \rightarrow (f_1 \vee y < 0)).
\end{aligned}$$

In addition, we assume no hints were provided, i.e. $\mathcal{H} \doteq \emptyset$. Let Ex and $\mathcal{H}$ be the inputs of our procedure. Alg. 1 at line 1 iterates over the candidate loops generated from Ex and $\mathcal{H}$. $\langle \boldsymbol{v}_0, loop\_r, loop\_t, H\rangle$. $loop\_r$ and $loop\_t$ are sequences of predicates over V and $V \cup V'$ respectively. The two sequences, together with H, describe the abstract loop. Instead, $\boldsymbol{v}_0$ is a state in the first region of $loop\_r$ reachable in Ex. Therefore, it is the last state of a finite path *prefix* of Ex that starts its initial states and ends in $\boldsymbol{v_0}$. We compute $\langle \boldsymbol{v}_0, loop\_r, loop\_t, H\rangle$ by employing a liveness-to-safety [9] transformation of Ex where the loop-back is identified in an abstract state. We then employ an unrolling of the transition relation in the style of Bounded Model Checking (BMC) [16] to enumerate concrete paths of Ex with such abstract loop-back. The stem of this concrete path corresponds to our *prefix*. $loop\_r$ and $loop\_t$ are obtained from the loop of the concrete path by computing an implicant for the unrolling of the transition relation of Ex. We then partition the predicates in the implicant depending on their index in the unrolling and whether they contain only current ($loop\_r$) or both current and next-state variables ($loop\_t$). Assume we are considering a BMC unrolling of 6 transitions of Ex and obtain the following path:

$$\begin{aligned}
0:&\quad f_0 \wedge f_1 \wedge pc = 3 \wedge x = 1 \wedge y = 1;\\
1:&\quad \neg f_0 \wedge \neg f_1 \wedge pc = 4 \wedge x = 1 \wedge y = 1;\\
2:&\quad f_0 \wedge \neg f_1 \wedge pc = 5 \wedge x = 1 \wedge y = -1;\\
3:&\quad f_0 \wedge f_1 \wedge pc = 3 \wedge x = 2 \wedge y = -1;\\
4:&\quad \neg f_0 \wedge \neg f_1 \wedge pc = 4 \wedge x = 2 \wedge y = -1;\\
5:&\quad \neg f_0 \wedge f_1 \wedge pc = 5 \wedge x = 2 \wedge y = 2;\\
6:&\quad f_0 \wedge f_1 \wedge pc = 3 \wedge x = 3 \wedge y = 2;
\end{aligned}$$

where the states with indexes 0 and 6 correspond to the same state in the abstract space defined by the predicates appearing in Ex. We use this path to compute an implicant for the formula $F(V_0) \wedge \bigwedge_{i=0}^{5} T(V_i, V_{i+1})$. The implicant is a conjunction of a subset of the atoms appearing in the formula such that it implies the formula itself. In addition, the path is a satisfying assignment also for the implicant. Each predicate in the unrolling depends either on a single

V_i or on $V_i \cup V_{i+1}$ for some i, hence the same holds for the predicates in the implicant. We partition the atoms of the implicant such that the predicates that depend only on V_i are in $\mathit{loop\_r}[i\%6]$ and those that depend on $V_i \cup V_{i+1}$ are placed in $\mathit{loop\_t}[i]$. The first and last state correspond to the same abstract region, hence their predicates are placed together into $\mathit{loop\_r}[0]$.

The computation above allows us to obtain the following. Since $\mathcal{H}$ is empty H is the trivial hint. The *prefix* contains a single state: $\mathit{prefix} \doteq [f_0 \wedge f_1 \wedge pc = 3 \wedge x = 1 \wedge y = 1]$, $\mathit{loop\_r}$ and $\mathit{loop\_t}$ have length 6 and each $\mathit{loop\_r}[i] \wedge \mathit{loop\_t}[i]$ underapproximates the transition relation T.

$$\begin{array}{lll}
 & \mathit{loop\_r} \doteq [& \mathit{loop\_t} \doteq [\\
0: & f_0 \wedge f_1 \wedge pc = 3 \wedge x^2 \geq xy, & \neg f_0' \wedge \neg f_1' \wedge pc' = 4 \wedge x' = x \wedge y' = y, \\
1: & \neg f_0 \wedge \neg f_1 \wedge pc = 4 \wedge y > 0, & f_0' \wedge \neg f_1' \wedge pc' = 5 \wedge x' = x, \\
2: & f_0 \wedge \neg f_1 \wedge pc = 5 \wedge y < 0, & f_0' \wedge f_1' \wedge pc' = 3 \wedge x' = x + 1 \wedge y' = y, \\
3: & f_0 \wedge f_1 \wedge pc = 3 \wedge x^2 \geq xy, & \neg f_0' \wedge \neg f_1' \wedge pc' = 4 \wedge x' = x \wedge y' = y, \\
4: & \neg f_0 \wedge \neg f_1 \wedge pc = 4 \wedge y < 0, & \neg f_0' \wedge f_1' \wedge pc' = 5 \wedge x' = x, \\
5: & \neg f_0 \wedge f_1 \wedge pc = 5 \wedge y > 0] & f_0' \wedge f_1' \wedge pc' = 3 \wedge x' = x + 1 \wedge y' = y]
\end{array}$$

We now search for a funnel-loop as a strengthening of this candidate loop. Notice that the candidate loop by itself is not sufficient. In fact, $\mathit{loop\_t}[1]$ does not constrain the next assignment of y', hence it does not guarantee that $y < 0$ holds in the next state as required by $\mathit{loop\_r}[2]$. Before building the funnel-loop template, we can perform some simplifications on the candidate loop to reduce the number of parameters introduced by the template and ease the presentation. First of all, notice that every step i in $\mathit{loop\_t}$ assigns to the variables f_0, f_1 and pc a constant value that corresponds to the one required by $\mathit{loop\_r}[(i+1)\%6]$. Therefore, for brevity, we will omit such constraints from the formulae in $\mathit{loop\_t}$ and focus our presentation on x and y. Moreover, many steps in $\mathit{loop\_t}$ require x or y to remain constant. Consider a step $t \doteq \mathit{loop\_t}[i]$ that requires y to be constant. We need t to map states in $r_s \doteq \mathit{loop\_r}[i]$ into $r_d \doteq \mathit{loop\_r}[(i+1)\%6]$. Therefore, if r_d requires y to be positive $(y > 0)$, then the same must hold in

r_s and vice-versa. We can exploit identity relations in *loop_t* to symbolically propagate constraints in *loop_r*. By employing these transformations we obtain the following:

$$\begin{array}{lll}
 & loop\_r \doteq [& loop\_t \doteq [\\
0: & f_0 \wedge f_1 \wedge pc = 3 \wedge x^2 \geq xy \wedge y > 0, & x' = x \wedge y' = y, \\
1: & \neg f_0 \wedge \neg f_1 \wedge pc = 4 \wedge y > 0, & x' = x, \\
2: & f_0 \wedge \neg f_1 \wedge pc = 5 \wedge y < 0, & x' = x + 1 \wedge y' = y, \\
3: & f_0 \wedge f_1 \wedge pc = 3 \wedge x^2 \geq xy \wedge y < 0, & x' = x \wedge y' = y, \\
4: & \neg f_0 \wedge \neg f_1 \wedge pc = 4 \wedge y < 0, & x' = x, \\
5: & \neg f_0 \wedge f_1 \wedge pc = 5 \wedge y > 0] & x' = x + 1 \wedge y' = y]
\end{array}$$

We now define a funnel-loop template of length 6 that can be generated by the procedure at line 3. For $i \in \{0, \dots, 5\}$, we define the i^{th} funnel of the template as a strengthening of $loop\_r[i]$ and $loop\_t[i]$. In the template we use symbols from the set $P \doteq \{p_i | i \in \mathbb{N}\}$, disjoint from V, as parameters. The parameters are variables for which we need to find an assignment such that the template corresponds to an actual funnel-loop. Notice that the steps in *loop_t* already prescribe functional assignments for all variables but for y at steps 1 and 4. For this reason, we introduce 2 parametric affine expressions to underapproximate the assignment to y'. In addition, we introduce parametric affine inequalities over x and y to strengthen the elements of *loop_r*. Also in this case we reduce the number of parameters we need to introduce by exploiting the functional assignments of *loop_t*. For $i \in \{0, 1 \dots, 5\}$, let $fnl_i \doteq \langle src_i, t_i, \mathbf{0}, dst_i \rangle$ be the i^{th} funnel of the template. We define each destination region dst_i as the set of states reachable from the previous source region when the ranking function is equal to the minimal element. Since we defined every ranking function to be always equal to the minimal element, we define each destination region as:

$$dst_i \doteq \exists V : src_i(V, P) \wedge t_i(V, V', P).$$

We define the source regions and transition relations as follows.

$$
\begin{aligned}
&src_0 \doteq loop\_r[0] \wedge p_6x + p_7y + p_8 \geq 0, && t_0 \doteq loop\_t[0],\\
&src_1 \doteq loop\_r[1] \wedge p_6x + p_7y + p_8 \geq 0, && t_1 \doteq loop\_t[1] \wedge y' = p_0x + p_1y + p_2,\\
&src_2 \doteq loop\_r[2] \wedge p_9x + p_{10}y + p_{11} + p_9 \geq 0, && t_2 \doteq loop\_t[2],\\
&src_3 \doteq loop\_r[3] \wedge p_9x + p_{10}y + p_{11} \geq 0, && t_3 \doteq loop\_t[3],\\
&src_4 \doteq loop\_r[4] \wedge p_9x + p_{10}y + p_{11} \geq 0, && t_4 \doteq loop\_t[4] \wedge y' = p_3x + p_4y + p_5,\\
&src_5 \doteq loop\_r[5] \wedge p_6x + p_7y + p_8 + p_6 \geq 0; && t_5 \doteq loop\_t[5].
\end{aligned}
$$

We introduced two parametric inequalities: $p_6x + p_7y + p_8 \geq 0$ at index 1 and $p_9x+p_{10}y+p_{11} \geq 0$ at index 4. Then, we propagated the inequalities backward exploiting the assignments to x and y of $loop\_t$. In particular, in $loop\_t[0]$ and $loop\_t[3]$ both x and y must remain constant. In $loop\_t[2]$ and $loop\_t[5]$, instead, y remains constant and x increases by 1. Therefore, $p_9x + p_{10}y + p_{11} \geq 0$ in src_3 implies that $p_9x + p_{10}y + p_{11} + p_9 \geq 0$ must hold in src_2 and similarly $p_6x + p_7y + p_8 \geq 0$ in src_0 implies $p_6x + p_7y + p_8 + p_6 \geq 0$ at src_5. We remark that exploiting the equalities in the transition relations is an optimisation we employ to reduce the number of parameters and has no effect on the correctness of the approach.

Now, we need to identify an assignment to the parameters $p_0, \ldots, p_{11}$ such that the funnel-loop template satisfies all hypotheses of Def. 1, Def. 2 and Th. 1. The procedure generates this synthesis problem at line 4 and it searches for a solution (assignment to the parameters) at line 5. The synthesis problem requires the funnel-loop to be reachable in Ex (FF.1), hence also not empty. We ensure this by requiring the first region of the funnel-loop to contain the last state of the prefix, hence the state $f_0, f_1, pc = 3, x = 1, y = 1$ must be in src_0. Then, the funnel-loop must never encounter a deadlock (F.1). This is true by construction of the transition relations of the funnels, because every t_i is left-total for every assignment to the parameters. We need the funnels to be correctly chained (FL.1, FL.2) and to underapproximate the transition relation T of Ex (FF.3). We defined the destination regions as the set of states reachable from the source region in one step. Therefore, we require the following to hold:

$$\exists P \forall V : src_i(V, P) \wedge t_i(V, V', P) \rightarrow src_{(i+1)\%6}(V', P) \wedge T(V, V')$$

Finally, every state in src_0 is a fair state, hence every path through the funnel-loop template is a fair path of Ex (FF.2).

The following assignment to the parameters satisfies all these requirements: $p_0 = 0, p_1 = -1, p_2 = 0, p_3 = 0, p_4 = -1, p_5 = 0, p_6 = 1, p_7 = -1, p_8 = 0, p_9 = 1, p_{10} = 1, p_{11} = 0$. We can substitute these values in the funnel-loop template and obtain the following funnel-loop.

$$\begin{aligned}
src_0 &\doteq loop\_r[0] \wedge x \geq y, & t_0 &\doteq loop\_t[0], \\
src_1 &\doteq loop\_r[1] \wedge x \geq y, & t_1 &\doteq loop\_t[1] \wedge y' = -y, \\
src_2 &\doteq loop\_r[2] \wedge x \geq -y, & t_2 &\doteq loop\_t[2], \\
src_3 &\doteq loop\_r[3] \wedge x \geq -y, & t_3 &\doteq loop\_t[3], \\
src_4 &\doteq loop\_r[4] \wedge x \geq -y, & t_4 &\doteq loop\_t[4] \wedge y' = -y, \\
src_5 &\doteq loop\_r[5] \wedge x \geq y; & t_5 &\doteq loop\_t[5].
\end{aligned}$$

Notice that in this process the parametric expressions allowed us to identify an underapproximation of the transition relation of Ex that toggles the sign of y instead of allowing any possible assignment. In addition, the parametric inequalities restricted the regions we obtained from the candidate loop to only the states in which $x \geq |y|$, hence ensuring that the loop condition $x^2 \geq xy$ of our example holds. In fact, $x^2 \geq xy$ is redundant in src_0 and src_3; it is implied by $x \geq y \wedge y > 0$ in the first region and by $x \geq -y \wedge y < 0$ in the second one. Therefore, this funnel-loop is equivalent to the one we defined in §5.3.

7.2. Candidate fair loops: representation and enumeration

We identify lasso-shaped paths in the abstract space built by the assignments to a finite set of predicates: two states that agree on the truth assignment for all such predicates correspond to the same abstract state. We then represent the fair loop as a sequence of transitions and regions (sets of states) that underapproximate the transition relation of M.

Given a fair transition system $M \doteq \langle V, I^M, T^M, F^M \rangle$ we describe a candidate fair loop of length n for M, associated with an E-component $H \doteq \langle V, I^H, T^H \rangle$ over regions $\mathcal{R} \doteq [R_i]_{i=0}^{n-1}$, assumptions $\mathcal{A} \doteq [A_i]_{i=0}^{n-1}$, ranking functions $\text{R}\textsc{f} \doteq [\text{R}\textsc{f}_i]_{i=0}^{n-1}$ and responsible for symbols $V^H \subseteq V$, as a sequence of regions $\mathit{loop\_r} \doteq [\mathit{loop\_r}_i(V)]_{i=0}^{n-1}$, transitions $\mathit{loop\_t} \doteq [\mathit{loop\_t}_i(V, V^{\neq H\prime})]_{i=0}^{n-2}$ and an initial state $\boldsymbol{v}_0$, where $V^{\neq H} \doteq V \setminus V^H$. Such that: (i) $\boldsymbol{v}_0 \models \mathit{loop\_r}_0 \wedge I^H$, (ii) $\boldsymbol{v}_0$ is reachable in M, (iii) the conjunction of a $\mathit{loop\_r}_i$ and the corresponding restricted region $R_i \wedge A_i$ underapproximates the fair states

$$\exists i \forall V : \mathit{loop\_r}_i \wedge R_i \wedge A_i \rightarrow F^M,$$

and (iv) for each step, the conjunction of $\mathit{loop\_t}_i$ and the transition relation T^H of H is an implicant for a transition in M

$$\begin{aligned} \forall i, V, V' \ : \ & (\mathit{loop\_r}_i \wedge R_i \wedge A_i \wedge \mathit{loop\_t}_i \wedge T^H \wedge \\ & ((\mathbf{0} < \text{R}\textsc{f}_i \wedge R'_i) \vee (\text{R}\textsc{f}_i = \mathbf{0} \wedge R'_{i+i}))) \rightarrow T^M. \end{aligned}$$

Without loss of generality, and to simplify the presentation, we assume the fair region to be the first one. The structure of a candidate loop resembles a funnel-loop. However, candidate loops are not guaranteed to satisfy all required hypotheses. In particular, the transitions $\mathit{loop\_t}_i \wedge T^H$ could admit deadlocks (Hyp. F.1) and they are not guaranteed to map every state in the previous region into some state in the following one (Hypotheses FL.1 and FL.2). In addition, H may not provide all the required ranking functions. For this reason, in order to identify a funnel-loop, we look for a strengthening of the candidate loop that also satisfies these conditions.

The enumeration of candidate loops and compositions is performed by Alg. 2. The procedure is based on Bounded Model Checking (BMC) [16], for the enumeration of candidate paths, and on the computation of an underapproximation of M for each path. The function ENCODE-L2S-FAIR-ABSTRACT-LOOP (line 1) encodes the search for a fair lasso-shaped path in the intersection of M and the composition of a subset of $\mathcal{H}$ into a reachability problem given by $\langle V, I, T, \mathit{bad} \rangle$. The problem requires us to identify paths over the variables V, starting in $I(V)$

Algorithm 2 GENERATE-CANDIDATE-LOOPS(M, $\mathcal{H}$)

```
    ▷ L2S encoding into reachability problem and $E$-component selection.
1:  $\langle V, I, T, bad \rangle \leftarrow$ ENCODE-L2S-FAIR-ABSTRACT-LOOP($M, \mathcal{H}$)
2:  for $k \in [0, 1, 2, \ldots]$ do                                ▷ BMC unrolling: k steps.
3:      $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k)$          ▷ BMC reachability.
4:      $\langle sat, model \rangle \leftarrow$ SMT-SOLVE($query$)              ▷ Find first path of length $k$.
5:      $refs \leftarrow []$                          ▷ Keep track of visited paths of length $k$.
6:      while $sat$ do        ▷ Generate all candidates from paths of same length.
7:          $H \leftarrow$ GET-CANDIDATE-COMPOSITION($model$)        ▷ Path selects hints.
8:          $\langle conflict \rangle \leftarrow$ GET-COMP-ERROR($H$)
9:          if $conflict \neq \bot$ then                       ▷ Learn incompatible transitions.
10:             $\langle V, I, T, bad \rangle \leftarrow$ REMOVE-CONFLICT($V, I, T, bad, conflict$)
11:         else                                            ▷ $H$ is valid $E$-component.
12:             $\langle loop\_r, loop\_t \rangle \leftarrow$ UNDERAPPROXIMATE($model, query, H$)
13:             $\langle is\_ranked, rf \rangle \leftarrow$ RANK-LOOP($loop\_r, loop\_t, H$)
14:             if $is\_ranked$ then                               ▷ Learn ranking function.
15:                 $\langle V, I, T, bad \rangle \leftarrow$ REMOVE-RANKED-LOOPS($V, I, T, bad, rf$)
16:             else ▷ Unable to find ranking function, could be nonterminating.
17:                 $prefix \leftarrow$ GET-PREFIX($model$)    ▷ Get stem of abstract lasso.
18:                 yield $\langle prefix, loop\_r, loop\_t, H \rangle$    ▷ Coroutine returns triples.
19:                 $refs.append(\neg(\bigwedge_{r \in loop\_r} r \wedge \bigwedge_{t \in loop\_t} t))$          ▷ Mark visited.
20:             end if
21:         end if
22:         $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k) \wedge \bigwedge_{ref \in refs} ref$
23:         $\langle sat, model \rangle \leftarrow$ SMT-SOLVE($query$)      ▷ Find next path of length $k$.
24:     end while
25: end for
```

and following the steps given by $T(V, V')$ that end in some state in $bad(V)$. We obtain this encoding via a liveness-to-safety [9] construction that transforms the problem of identifying an abstract lasso into a reachability problem. The loop-back state is identified in the abstract space defined by the predicates in the E-components and in the transition relation and fairness condition of M. The last state and the loop-back state of the abstract lasso must agree on the truth assignment of all such predicates, hence they may not be the very same assignment. In the encoding, a set of fresh Boolean variables selects the E-components to be considered, and the path must be such that at most one ranking function decreases at a time. We then rely on a SMT-solver to identify fair lasso paths of increasing length k (line 2), as done for the abstract liveness-to-safety algorithm of [11]. The models of this BMC unrolling describe a path in the language of both M and the composition of a subset of the E-components in $\mathcal{H}$. If $\mathcal{H}$ is empty or none of the hints is selected, GET-CANDIDATE-COMPOSITION (line 7) returns the trivial hint H of length equal to the number of states in the abstract lasso. Instead, if some hints are selected, H is given by their composition projected over the ordered sequence of regions visited by the path. The selected E-components might not be compatible, for this reason, after extracting the candidate composition at line 7 from the BMC model, GET-COMP-ERROR (line 8) checks if each transition in the composition is compatible (the trivial hint is trivally compatible). If this is not the case, a conflict predicate representing the transitions that are not compatible is used by REMOVE-CONFLICT to refine the reachability problem $\langle V, I, T, \mathit{bad}\rangle$ such that we avoid generating the same conflict again. If H is a valid E-component the function UNDERAPPROXIMATE (line 12) computes two sequences of $n-1$ formulae: $\mathit{loop\_r} \doteq [\mathit{loop\_r}_i(V)]_{i=0}^{n-2}$ and $\mathit{loop\_t} \doteq [\mathit{loop\_t}_i(V, V')]_{i=0}^{n-2}$ such that each $\mathit{loop\_r}_i \wedge \mathit{loop\_t}_i$, together with H, underapproximates the transition relation of M. The function computes an implicant for the formula *query* such that the assignments of the lasso described by *model* satisfy both formulae. We then partition, for each step i, the predicates in the implicant into two sets. All predicates containing only symbols of V at step i are in $\mathit{loop\_r}_i$, while the predicates containing symbols from $V \cup V'$ at step i are in $\mathit{loop\_t}_i$. Therefore, we split the predicates used to describe the regions from the ones that constrain the transitions. We use $\mathit{loop\_r}_i$ and $\mathit{loop\_t}_i$ as formulae meaning the conjunction of all the predicates in the set and they, together with H, describe the candidate fair loop.

Then, at line 13, we try to synthesise a ranking function for such candidate loop via the method RANK-LOOP. In the literature there are many approaches for the synthesis of ranking functions [17, 18, 19], here we simply assume we are given a method that returns the representation of a ranking function rf proving the termination of a candidate loop. If the procedure succeeds in identifying a ranking function, the reachability problem $\langle V, I, T, bad\rangle$ is refined such that we avoid enumerating other loops ranked by the same function, as described in [11]. This is achieved by calling REMOVE-RANKED-LOOPS at line 15). Otherwise, at line 17, GET-PREFIX extracts from *model* the prefix of the loop; i.e. the path of M ending in the the loop-back state. The prefix represents a witness for the reachability of the first region of the candidate loop in M and the procedure returns it together with the current candidate loop, at line 18. If no candidate loop of length k exists, we clear the list of refinements and enumerate the candidate loops of length $k+1$.

Example. We now provide a brief example on the computation of the under-approximation of M described by *loop_r* and *loop_t*. Assume the transition relation of M is $T \doteq (a \leq 1 \rightarrow b' > b) \wedge (a \geq 2 \rightarrow b' < b)$, and the loop described by *model* is given by the assignments $a_0 = 1, b_0 = 0$, $a_1 = 2, b_1 = 1$ and $a_2 = 0, b_2 = 0$. Three steps of M are represented by the formula $T(V_0, V_1) \wedge T(V_1, V_2)$. An implicant for such formula satisfied by *model* is given by $\{a_0 \leq 1, b_1 > b_0, a_1 \geq 2, b_2 < b_1\}$. Such an implicant can be obtained, for example, by applying the techniques presented in [20, 21]. Finally, we partition this set into *loop_r* and *loop_t* by defining their components as follows: $loop\_r_0 \doteq a \leq 1$, $loop\_t_0 \doteq b' > b$, $loop\_r_1 \doteq a_1 \geq 2$ and $loop\_t_1 \doteq b' < b$.

7.3. Funnel-loop templates

We call funnel-loop template a candidate funnel-loop whose predicates contain symbols of both V and a set of parameters P (P and V are disjoint). For each candidate fair loop we generate a sequence of such templates and try to identify an assignment to the symbols P such that by replacing them with the identified values we obtain a funnel-loop satisfying all the required hypotheses. In the following, the function called NEW-PARAM-EXPR generates expressions over the symbols V and the parameters P, e.g. affine linear functions $p_0 + \sum_{i=1}^{|V|} p_i v_i$, where $|V|$ is the number of symbols in V and for all i, $p_i \in P$ and $v_i \in V$. The function introduces as many parameters as necessary to generate the required expressions.

Alg. 3 shows the procedure we use to generate funnel-loop templates from a candidate loop. We generate templates of the same length as the candidate loop. Function HEURISTIC-PICK-NUM-INEQS (line 1) selects a list of natural numbers to be used to generate the funnel-loop templates. Each number corresponds to the amount of parametric inequalities added to each region of the candidate loop to define the corresponding source region of a funnel template (line 7). The higher the number the more freedom will the template have in shrinking the regions, but in the search problem we will have more parameters and a larger space to explore. Notice that, since the first region of the candidate loop is fair by construction, then the last destination region in the funnel-loop template will be fair and Hyp. FF.2 holds. We create the i^{th} funnel of the funnel-loop template (lines 6–25) as a restriction of the conjunction of the i^{th} region and transition of the candidate loop. In addition, the only nondeterministic component in t is given by the transition relation of H. All other components of the transition relation t of the funnel are a deterministic functional assignment as follows. Let V^H be the symbols for which H is responsible. For each symbol $v_{i+1} \in V_{i+1} \setminus V^H_{i+1}$, if $loop\_t_i$ already contains a functional assignment for v_{i+1}, then we use that (line 17). Otherwise, we generate a functional assignment for v_{i+1} as a parametric expression over the symbols in V_i (line 19). The resulting transition relation is total and Hyp. F.1 holds. We define the destination region of a funnel implicitly as the set of states reachable in one step from $S \wedge \textsc{Rf} = \mathbf{0}$ (line 23), hence Hyp. F.4 holds by construction. Finally, the procedure returns the funnel-loop template associated with the list of parametric funnels and initial state $\boldsymbol{v}_0$. We will ensure that $\boldsymbol{v}_0$ is in the first source region of the funnel-loop.

Algorithm 3 GENERATE-TEMPLATES($\boldsymbol{v}_0, loop\_r, loop\_t, H$)

```
1: $ineqs \leftarrow$ HEURISTIC-PICK-NUM-INEQS($loop\_r, loop\_t, H$)
2: $\langle V^H, I^H, T^H, \mathcal{R}, \mathcal{A}, \textsc{Rf}\rangle \leftarrow H$    ▷ Get components defining $H$.
3: for $ineq \in ineqs$ do    ▷ Use $ineq$ parametric inequalities in regions.
4:     $n \leftarrow \mathbf{len}(loop\_r)$    ▷ Length of template + 1: loop-back region.
5:     $funnels \leftarrow []$    ▷ List of funnels for funnel-loop template.
6:     for $i \in [0..n-2]$ do    ▷ Create $i^{\text{th}}$ funnel: $\langle src, t, rf, dst\rangle$.
7:         $src \leftarrow loop\_r[i] \wedge \mathcal{R}[i] \wedge \mathcal{A}[i] \wedge \bigwedge_{j=0}^{ineq-1}$ NEW-PARAM-EXPR$(V) \geq 0$
8:         if $\exists V : \mathbf{0} < \textsc{Rf}[i](V)$ then
9:             $rf \leftarrow \textsc{Rf}[i]$    ▷ $H$ defines ranking function.
10:        else
11:            $rf \leftarrow$ NEW-PARAM-EXPR$(V)$    ▷ Parametric ranking function.
12:        end if
13:        $t \leftarrow \mathcal{R}[i] \wedge \mathcal{A}[i]$    ▷ Transition of $H$ in $i^{\text{th}}$ region.
14:        $t \leftarrow t \wedge T^H \wedge ((\mathbf{0} < rf \wedge \mathcal{R}[i]' \wedge rf' \leq rf) \vee (rf = \mathbf{0} \wedge \mathcal{R}[i+1]'))$
15:        for $v_{i+1} \in V_{i+1} \setminus V^H_{i+1}$ do    ▷ Add functional assign for $v_{i+1}$ in $t$
16:            if $v_{i+1} = f(V_i) \in loop\_t[i]$ for some function $f$ then
17:                $t \leftarrow t \wedge v_{i+1} = f(V_i)$    ▷ Functional assignment in candidate.
18:            else
19:                $t \leftarrow t \wedge v_{i+1} =$ NEW-PARAM-EXPR$(V_i)$    ▷ Create new expr.
20:            end if
21:        end for
22:        $P \leftarrow$ COLLECT-PARAMETERS$(src, rf, t)$    ▷ Params in current funnel.
23:        $dst(V', P) \leftarrow \exists V \ : \ src(V, P) \wedge rf(V, P) = \mathbf{0} \wedge t(V, V', P)$
24:        $funnels.append(\textsc{Funnel}(src, t, rf, dst))$
25:    end for
26:    yield FUNNEL-LOOP$(funnels, \boldsymbol{v}_0)$    ▷ Coroutine returns templates.
27: end for
```

Therefore, since $\boldsymbol{v}_0$ is reachable in M, Hyp. FF.1 holds.

Example. We now provide an example to illustrate how a funnel is generated in the lines from 7 to 24. In this example we assume the following: $V \doteq \{a, b, c\}$ is a set of real-valued symbols; NEW-PARAM-EXPR generates affine linear expressions over V and a set of parameters $P \doteq \{p_i\}_{i \in \mathbb{N}}$; we are constructing a funnel-loop template adding a single predicate to shrink the region ($ineq = 1$); $loop\_r[i] \doteq b < c$; $loop\_t[i] \doteq c' = c \land b' > b + a \land b' > c$ and the hint H responsible for $\{a\}$ has the following components: $\mathcal{R}[i] \doteq a > 0$, $\mathcal{R}[i+1] \doteq a > 0$, $\mathcal{A}[i] \doteq \top$, $\textsc{Rf}[i] \doteq \mathbf{0}$ and $T^H \doteq a' > a$.

For simplicity, we defined P as an infinite set. However, in this example we will use 12 parameters $\{p_i\}_{i=0}^{11}$; we will introduce 3 affine parametric expressions each of which requires 4 parameters. The first expression represents an additional inequality for the region, the second one is used to represent the ranking function, and the last one corresponds to the functional assignment of b' in the transition relation.

Line 7 defines the source region *src* of the funnel as the conjunction of the $loop\_r[i]$, the restricted region of H and, since $ineq = 1$ it introduces a single parametric predicate: $p_0 + p_1 a + p_2 b + p_3 c \geq 0$.

$$src(V, P) \doteq b < c \land a > 0 \land p_0 + p_1 a + p_2 b + p_3 c \geq 0.$$

The condition at line 8 is false since the ranking function provided by H is always equal to $\mathbf{0}$. The procedure then executes line 11, which introduces a new parametric expression to represent the ranking function:

$$rf(V, P) \doteq p_4 + p_5 a + p_6 b + p_7 c.$$

We implicitly consider the function equal to the minimal element $\mathbf{0}$ if $rf(V, P) \leq 0$. Then, line 14 initialises t from the transition relation of H as:

$$t \doteq a > 0 \land a' > a \land ((rf(V, P) \leq 0 \land a' > 0) \ \lor \\ (0 < rf(V, P) \land a' > 0 \land rf(V', P) \leq rf(V, P))).$$

The loop starting at line 15 iterates over the symbols in $\{b, c\}$. Consider first the symbol c, in $loop\_t[i]$ we find the equality $c' = c$, hence the condition at line 16 holds and the equality is added to t as a conjunct. Consider now the symbol b, $loop\_t[i]$ prescribes no equality for b', hence a new parametric expression is introduced and added to t at line 19, let such equality be $b' = p_8 + p_9 a + p_{10} b + p_{11} c$. Therefore, the final t is as follows:

$$t \doteq a > 0 \wedge a' > a \wedge ((0 < rf(V,P) \wedge a' > 0 \wedge rf(V',P) \leq rf(V,P)) \vee \\ (rf(V,P) \leq 0 \wedge a' > 0)) \wedge c' = c \wedge b' = p_8 + p_9 a + p_{10} b + p_{11} c.$$

Finally, dst is defined as the set of states that admit a predecessor through t in src with $rf = \mathbf{0}$:

$$dst(a',b',c',P) \doteq \exists a,b,c : src(a,b,c,P) \wedge rf(a,b,c,P) \leq 0 \wedge t(a,b,c,a',b',c',P).$$

7.4. Funnel-loop synthesis problem

We now describe the $\exists\forall$ quantified formula that corresponds to the synthesis problem of a funnel-loop template. Alg. 1 computes this formula for every funnel-loop template *template* via the method *ef_constraints* at line 4. We search for an assignment to the parameters P of the funnel-loop template such that by replacing them with the identified values we obtain a funnel-loop that satisfies all hypotheses of Defs. 1, 2 and of Th. 1. In the hypotheses, for every funnel $fnl_i \doteq \langle S_i, T_i, D_i, \mathrm{RF}_i \rangle$, we replace each destination region D_i with the quantified formula:

$$D_i(V') \doteq \exists V : S_i(V) \wedge \mathrm{RF}_i(V) = \mathbf{0} \wedge T_i(V,V'). \tag{1}$$

Every instance of the funnel-loop template must contain a fair region since $loop\_r_0$ is a subset of the fair states and S_0, by construction, underapproximates $loop\_r_0$. We ensure that Hyp. FF.1 holds by requiring that $\boldsymbol{v}_0$ is in the source region of the first funnel fnl_0 with the constraint:

$$\exists P \ : \ S_0(\boldsymbol{v}_0, P). \tag{2}$$

Hyp. F.1 holds by construction, since the next region implies the assumptions required by the E-component. Therefore, the transition relation of the

E-component must always allow for a successor for all assignments to the $V^{\neq H'}$. In addition, the other components of the transition relation of the funnel describe a functional assignment to the $V^{\neq H'}$ without any circular dependency. Hyp. F.4 holds since we implicitly defined the destination region of each funnel fnl_i as the set of states reachable in one step from $S_i \wedge \textsc{Rf}_i = \mathbf{0}$. Then, we ensure that every instantiation of every funnel template fnl_i in the funnel-loop template satisfies hypotheses F.2 and F.3 by requiring that the following hold:

$$\exists P\ \forall V, V' : (S_i(V,P) \wedge \mathbf{0} < \textsc{Rf}_i(V,P) \wedge T_i(V,V',P)) \to S_i(V',P); \tag{3}$$

$$\exists P\ \forall V, V' : (S_i(V,P) \wedge \mathbf{0} < \textsc{Rf}_i(V,P) \wedge T_i(V,V',P)) \to \textsc{Rf}_i(V',P) < \textsc{Rf}_i(V,P). \tag{4}$$

The funnels must be correctly chained for hypotheses FL.1 and FL.2 to hold. Notice that in these formulae are implications whose left-hand-side is D_i and we bring the existential quantifier out in front of the formula as a universal quantifier. For Hyp. FL.1 to hold we require every two consecutive funnel templates fnl_i and fnl_{i+1} in the funnel-loop template to satisfy the following:

$$\exists P\ \forall V, V' : (S_i(V,P) \wedge \textsc{Rf}_i(V,P) = \mathbf{0} \wedge T_i(V,V',P)) \to S_{i+1}(V',P). \tag{5}$$

Similarly, considering the first and last funnels fnl_0 and fnl_{n-1}, for Hyp. FL.2 we require:

$$\exists P\ \forall V, V' : (S_{n-1}(V,P) \wedge \textsc{Rf}_{n-1}(V,P) = \mathbf{0} \wedge T_{n-1}(V,V',P)) \to S_0(V',P). \tag{6}$$

This ensures that D_{n-1} is a subset of S_0. We have observed above that S_0 contains only fair states, hence FF.2 holds. Finally, we require each funnel-loop instance to underapproximate M (Hyp. FF.3) by requiring the following to hold for every funnel fnl:

$$\exists P\ \forall V, V'\ :\ S(V,P) \wedge T(V,V',P) \to T^M(V,V'). \tag{7}$$

The final synthesis problem is then given by the conjunction of all the constraints (1)–(7). A solution for this problem is a model that assigns a value to each symbol in P such that the formulae hold for all possible assignments to the symbols in $V \cup V'$. From one such model we can generate a concrete funnel-loop by substituting the parameters P with their assignment.

8. Related work

Most of the literature in verification of temporal properties of infinite-state transition systems, hybrid automata and termination analysis focuses on the universal case, while the existential one has received relatively little attention.

Most closely related are the works concerned with proving *program non-termination*. The works [22] and [5] are based on the notion of closed recurrence set, that corresponds to proving the non-termination of a relation. Instead, the works [23] and [24] search for non-terminating executions via a sequence of safety queries. Other approaches look for specific classes of programs ([25] and [26] prove the decidability of termination for linear loops over the integers), or specific non-termination arguments (in [27] non-termination is seen as the sum of geometric series). However, none of these works deals with fairness and they rely on the existence of a control flow graph, whereas we work at the level of transition system.

The work [28] reduces the verification of the universal fragment of CTL on a infinite-state transition system to the problem of deciding whether a program always returns true. The approach can be applied also on LTL properties by relying on a reduction based on prophecy variables and it relies on some off-the-shelf tool for the analysis of the program. Therefore, its capability of proving or identifying a counterexample for some property depends on the ones of the considered underlying tool.

The work [29] explicitly deals with fairness for infinite-state programs and supports full CTL*; it is able to deal with existential properties and to provide fair paths as witnesses. The approach focuses on programs manipulating integer variables, with an explicit control-flow graph, rather than more general symbolic transition systems expressed over different theories (including real arithmetic). Another approach supporting full CTL* is proposed in [30]. The work presents a model checking algorithm for the verification of CTL* on finite-state systems and a deductive proof system for CTL* on infinite-state systems. In the first case the authors reduce the verification of CTL* properties to the verification of properties without temporal operators and a single fair path quantifier in front of the formula. To the best of our knowledge there is no generalisation of this algorithm, first reported in [31] and then also in [32], to the infinite state setting. The rules presented in the second case have been exploited in [33] to implement a procedure for the verification of CTL properties, while our objective is the falsification of LTL properties. Moreover, in these settings ([29], [30]) there is no notion of non-zenoness.

The works on *timed automata* are less relevant: although the concrete system may exhibit no lasso-shaped witnesses, due to the divergence of clocks, the problem is decidable, and lasso-shaped counterexamples exist in finite bisimulating abstractions. This view is adopted, for example, in Uppaal [34]. Other tools directly search for non lasso-shaped counterexamples, but the proposed techniques are specific for the setting of timed automata [35, 13] and lack the generality of the method proposed in this paper.

Our approach can be applied also to *hybrid systems*. Most of the works in this context are concerned with the verification of safety properties [36]. Instead, we deal with the falsification of general LTL, liveness properties. The works [37] and [38] propose a general approach for the verification of LTL properties on such systems. However, they can only identify lasso-shaped counterexamples and lack the generality of the approach we present in this work. Other approaches consider only particular types of liveness properties or impose additional restrictions on the model. The technique presented in [39] considers only stability properties and [40] falsifies properties under robustness assumptions, while [41] considers robust discrete time systems. In [42] the authors propose a technique to falsify LTL properties via randomised search, however it is restricted to safety LTL and does not consider *Zeno* paths.

Inductive Reachability Witness (IRW), defined in [15], is a structure roughly equivalent to our definition of funnel. [15] proposes to identify a single IRW as a witness for reachability queries in imperative programs over real variables: hence as a compact representation of a finite path. Instead, we look for a sequence of funnels, in the form of a funnel-loop, that represents an infinite path for an infinite-state fair transition system.

Finally, the verification conditions we identify in this work for the search of a funnel-loop can be expressed in the form of *existentially-quantified constrained Horn-like clauses* (E-CHCs) [33].[7] E-CHCs are an extension of *constrained Horn clauses* (CHCs) [43, 44, 45] with existential quantifiers. The two formalisms have been proposed as means to decouple the definition of verification problems from the actual solving algorithm. This enables the separation of the proof methodology from the procedures used to address the problem. Unfortunately, we were not able to obtain any tool capable of identifying solutions to E-CHCs, hence we could not investigate this direction any further.

9. Experimental Evaluation

This section first presents our implementation of the approach (9.1), then describes the benchmarks we used (9.2) and briefly presents the other state-of-the-art tools we considered (9.3), finally it reports the setup, the results and the discussion of the experimental evaluation we performed (9.4).

9.1. Implementation

We have implemented these procedures in a prototype, called F3[8] (for FindFairFunnel), written in Python. F3 uses MathSAT5 [46] and Z3 [47] as underlying SMT engines, interacting with them through pysmt [48]. SMT-solvers sometimes take a very long time on a single query. For this reason we associate a timeout to each call to SMT-solve. If the solver is unable to answer within

[7] Appendix A reports an encoding for the funnel-loop search problem in E-CHCs and proves it to be both sound and complete.

[8] the tool and the benchmarks can be downloaded from https://github.com/enmag/F3

the given time F3 assumes unknown as result and continues. F3 takes as input a transition system M, a fairness condition F and a possibly empty set of E-components $\mathcal{H}$, and tries to identify a funnel that proves that M admits at least one path that visits F infinitely-often. We then employ the usual tableau construction to support LTL specifications via reduction to the previous case. In order to support timed systems, we use the product construction described in [38] to remove all Zeno-paths of the model. F3 enumerates funnel templates in increasing order of complexity. By default, F3 considers a minimum of 0 and a maximum of 2 inequalities in the implementation of HEURISTIC-PICK-NUM-INEQS of Alg. 3. F3 considers only simple ranking functions corresponding to the PR-ranking template described in [17] which are simple affine linear functions. Such ranking functions are used in the definition of the funnel templates and when trying to identify a ranking function for a candidate abstract loop in Alg. 2. In addition, we only synthesise predicates in the form of affine linear equalities or inequalities; the implementation of the function NEW-PARAMETRIC-EXPR in F3 generates affine linear expressions. An important optimization is that F3 generates ranking function templates (line 11 of Alg. 3) only when it finds a pair of abstract states that prescribe the same assignment to the Boolean variables of M; if the abstract states differ in their Boolean variables, rf is simply set to the constant $\mathbf{0}$. This avoids the introduction of unnecessary parameters for funnels which do not need an explicit ranking function. F3 solves the parameter synthesis problem described in Sec. 7 via a combination of the EF-SMT procedure of [49] and the application of Motzkin's transposition theorem [50] to reduce the problem into a purely existentially-quantified one which can then be solved via standard quantifier-free SMT reasoning: we first try to apply EF-SMT, and resort to the elimination of universal quantifiers only if this fails to provide a definite answer. Finally, when applying the Motzkin's transposition theorem, F3 replaces non-linear terms with fresh symbols, in order to obtain a linear system. This simple way of handling non-linearities has the benefit of being very easy to implement; on the other hand, however, it can produce very coarse approximations, which can prevent F3 from finding counterexamples in cases where non-linearities play a significant role. A possible approach to handle non-linearities in a more precise manner is described in [15].

9.2. Benchmarks

In order to evaluate the effectiveness of our method, we have evaluated F3 on a wide range of benchmarks coming from different domains, from software (non)termination to timed automata and infinite-state symbolic transition systems. More specifically, we considered a total of 455 benchmarks, divided into 6 categories:

LS consists of 52 nonterminating linear software benchmarks taken from the C programs of the software termination competition [51];

NS contains 30 nonlinear software programs, of which 29 have been taken from [5] and one we defined;

ITS are 70 LTL falsification problems on infinite-state systems; 2 of such problems are proof obligations generated in the verification of a contract-based design, 29 come from the scaling to up to 30 processes of a model of the bakery mutual exclusion protocol in which we introduced a bug, other 29 come from the scaling to up to 30 processes of a semaphore-based synchronisation protocol, and the last 10 are instances we created;

TA contains 174 LTL falsification problems on timed automata; we consider 6 different protocols taken from [52] (*critical*, *csma*, *fddi*, *fischer*, *lynch* and *train*) and scale each of them from 1 to 30 processes;

TTS consists of 120 LTL falsification problems on timed transition systems, of which 116 come from the scaling from 1 to 30 processes of 4 protocols (inspired by the *csma*, *fischer*, *lynch* and *token ring* protocols), and 4 are handcrafted instances. The 4 protocols are a subset of the ones we considered in the TA instances. However, in this case we have extended them to obtain models that cannot be represented as timed automata. For the *csma* protocol we introduced an adaptive backoff time for each process that increases every time a station encounters a collision and decreases each time it successfully communicates the whole message. We extended the *fischer* and *lynch* protocols by allowing each process to propose a wait time, then the actual waiting time used to ensure mutual exclusion is the maximum of the proposed values. Finally, in the *token ring* protocol we added a stopwatch variable that keeps track of the total amount of time spent while transmitting and we ask to verify whether such value is bounded by 10 subject to a fairness assumption on the token manager of the protocol.

HS are 9 LTL falsification problems on hybrid systems (encoded as nonlinear infinite-state transition systems), 5 of which have been taken from the ARCH competition [53] and 4 are models of a bouncing ball which differ on the behaviour of the bounce.

F3 only handles symbolic transition systems, and not software programs; therefore, we have encoded the software benchmarks as infinite-state transition systems by introducing an explicit program counter as state variable. Moreover, since F3 only supports systems with Boolean, integer and real variables, we have not considered programs that involve recursion or dynamic memory allocation.

9.3. Competitor tools

We compare F3 with the following state-of-the-art tools: Anant [5], AProVe [54], DiVinE3 [55], iRankFinder [56], MITLBMC [57], nuXmv [13], T2 [58], Ultimate [59] and Uppaal [60]. Unfortunately we could not obtain the software described in [33] to solve E-CHC problems. Most of the other tools are not able to handle all the benchmarks we have considered. Therefore, we limit their application as follows:

- we ran Anant, AProVe, iRankFinder and T2 only on the software nontermination problems (LS and NS groups);
- we ran DiVinE3, MITLBMC and Uppaal only on the timed automata (TA) benchmarks; moreover, since Uppaal supports only a fragment of

Table 1: Summary of experimental results (number of solved instances per benchmark family).

Benchmark family	**F3 (no hints)**	**Anant**	**AProVe**	**DiVinE3**	**iRankFinder**	**MITLBMC**	**nuXmv**	**T2**	**Ultimate**	**Uppaal**
LS (52)	**52**	38	43	–	39	–	28	38	49	–
NS (30)	**30**	25	5	–	6	–	14	2	–	–
ITS (70)	**67**	–	–	–	–	–	4	–	8	–
TA (174)	130	–	–	43	–	**151**	90	–	0	103
TTS (120)	**50**	–	–	–	–	–	8	–	1	–
HS (9)	**4**	–	–	–	–	–	0	–	–	–
Total (455)	**333**	63	48	43	45	151	144	40	58	103

Entries marked with "–" denote that the tool cannot handle the given benchmarks.

LTL which is not sufficient to express the properties of the *fischer* and *lynch* benchmarks, we could run it only on 116 of the 174 TA instances;

- as Ultimate doesn't support non-linear arithmetic, we didn't run it on the NS family. Moreover, since it supports LTL specifications but works on programs rather than transition systems, we translated the benchmarks to LTL verification problems on software programs, using the same approach described in [11].

- nuXmv is the only other tool (besides F3) that supports all the benchmarks. Our focus is falsification of universal properties (or dually verification of existential ones), hence we ran nuXmv using only its BMC engine. The other algorithms available in nuXmv have no additional falsification capabilities and also try to verify the property.

9.4. Evaluation

We executed each tool on the corresponding benchmarks on a machine running Ubuntu 20.04 equipped with an Intel(R) Xeon(R) Gold 6226R 2.90 GHz

CPU, using a 1 hour timeout and a memory limit of 30 GB for each benchmark. A summary of the evaluation results is reported in Table 1. We run F3 on those benchmarks without providing any hint and the table shows, for each tool, the number of solved instances in each benchmark family. When a tool is not applicable to a specific family, this is marked with "-". From the table, we can see that F3 solved the highest number of instances overall and also the highest number of instances in all categories with the exception of timed automata. In this category F3 is outperformed only by MITLBMC, which implements a technique explicitly developed for timed automata. This demonstrates the generality of our approach, although (unsurprisingly) it is possible to define more efficient procedures to target specific classes of problems. F3 successfully identifies a fair path in all nonlinear software benchmarks and also in 4 of the hybrid (nonlinear) systems. Therefore, while being coarse-grained, the approximation of the nonlinear terms used by F3 appears to be sufficient in these cases. Finally, we should remark that the competitor tools (with the exception of MITLBMC and nuXmv in BMC mode) are also able to prove that a universal property holds, whereas F3 can only find counterexamples. On the other hand, however, our techniques can be easily integrated with approaches focusing on proving properties, such as [11, 38].

We then considered the 5 hybrid benchmarks that F3 failed to solve without hints. In 4 cases the definition of a single hint is sufficient to allow F3 to identify a fair path. The remaining benchmark is a handcrafted one representing a bouncing ball such that the interval of time between consecutive bounces follows the harmonic series and the tool is required to identify a non-Zeno path in which the ball keeps bouncing forever. We know that the harmonic series diverges, hence such a path exists. However, the path does not have the finite-variability property, often assumed in real-time temporal logics (e.g. [61, 62]); there is no bound on the number of times predicates change truth assignment for any finite interval of time: there is no lower bound on the time between two bounces. In addition, the absence of such bounds hinders the definition of simple ranking functions, since they require a minimum constant progress at every transition. We remark that the HS instances are the most complex ones, they involve both nonlinearities and timing constraints. The definition of the hints for such complex systems requires in depth analysis of each model and also an understanding of the features that the automated procedure could struggle to analyse. However, the integration of the hints within an automatic procedure allows the user to focus on the few hardest components of the model, while relying on the automated procedure to analyse the relatively simpler ones. Therefore, it provides an additional possibility to be explored to analyse the system before resorting to a purely manual inspection. Our experiments, while relatively limited in number, showed this approach to be viable allowing the procedure to identify 4 additional counterexamples on complex instances that no other tool managed to address successfully.

We conclude with some general observations about F3. F3 identifies funnel-loops by trying to instantiate a number of templates. As in every template-based approach, this implies that it will fail to identify funnel-loops that do not match the considered templates. For example, F3 generates the templates by strengthening the candidate loops with affine expressions and inequalities, hence it will fail to identify funnel-loops that require the procedure to identify nonlinear assignments or constraints. In our experiments this issue has been mitigated by the fact that the candidate loop itself might provide the necessary nonlinear terms, hence F3 does not need to synthesise them. F3 employs symbolic reasoning and inherits the instability typical of this kind of techniques that deal with undecidable problems. The execution time of F3 is greatly affected by the order in which the candidate loops are explored. For each candidate loop for which it fails to identify a ranking function, F3 generates and tries to instantiate a number of funnel-loop templates. The number of these templates can be relatively large and, in our experiments, F3 spent most of the time in trying to instantiate them. For this reason, the execution time of F3 might change significantly depending on the order in which the SMT-solvers identify candidate loops. F3 tries to mitigate this problem by analysing the templates in increasing order of complexity and by applying heuristics normalizations on the expressions before calling the SMT-solver. In principle each funnel-loop template can be analysed independently from the others and performing such tasks in parallel could mitigate this issue; in addition, one could also analyse different candidate loops in parallel. However, we did not explore this possibility and our prototype does not employ any kind of parallelism.

10. Conclusions

In this work we presented an approach to automatically verify existential properties on infinite-state fair transition systems which can also benefit from some user-defined hints. The witness for the existential property is given as a sequence of funnels and can represent paths that do not have a lasso-shape structure. We evaluated a prototype implementation of the approach on a wide variety of benchmarks. The prototype is effective and able to address verification tasks successfully in many different domains. However, there are still some classes of problems that exhibit behaviours that are outside the scope of our prototype, as we have seen in the case of the harmonic bouncing ball.

In the future, we plan to improve the procedure by automating the system decomposition and by investigating different heuristics for the selection of funnel-loop templates and to better exploit the system decomposition. Another interesting direction is to improve the support for nonlinear expressions, e.g. it should be possible to integrate the technique presented in [15] in our procedure. Finally, we plan to integrate our procedure with dual approaches used to verify LTL properties.

References

[1] A. Cimatti, A. Griggio, E. Magnago, Ltl falsification in infinite-state systems, Information and Computation (2022) 104977doi:https://doi.org/10.1016/j.ic.2022.104977.
URL https://www.sciencedirect.com/science/article/pii/S0890540122001328

[2] A. Cimatti, A. Griggio, E. Magnago, Proving the existence of fair paths in infinite-state systems, in: F. Henglein, S. Shoham, Y. Vizel (Eds.), Verification, Model Checking, and Abstract Interpretation - 22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17-19, 2021, Proceedings, Vol. 12597 of Lecture Notes in Computer Science, Springer, 2021, pp. 104–126. doi:10.1007/978-3-030-67067-2_6.

[3] A. Cimatti, A. Griggio, E. Magnago, Automatic discovery of fair paths in infinite-state transition systems, in: Z. Hou, V. Ganesh (Eds.), Automated Technology for Verification and Analysis - 19th International Symposium, ATVA 2021, Gold Coast, QLD, Australia, October 18-22, 2021, Proceedings, Vol. 12971 of Lecture Notes in Computer Science, Springer, 2021, pp. 32–47. doi:10.1007/978-3-030-88885-5_3.

[4] D. Giannakopoulou, K. S. Namjoshi, C. S. Pasareanu, Compositional reasoning, in: E. M. Clarke, T. A. Henzinger, H. Veith, R. Bloem (Eds.), Handbook of Model Checking, Springer, 2018, pp. 345–383. doi:10.1007/978-3-319-10575-8_12.

[5] B. Cook, C. Fuhs, K. Nimkar, P. W. O'Hearn, Disproving termination with overapproximation, in: Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21-24, 2014, IEEE, 2014, pp. 67–74. doi:10.1109/FMCAD.2014.6987597.
URL http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6975680

[6] A. Pnueli, The temporal logic of programs, in: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977, IEEE Computer Society, 1977, pp. 46–57. doi:10.1109/SFCS.1977.32.

[7] M. Y. Vardi, An automata-theoretic approach to linear temporal logic, in: Banff Higher Order Workshop, Vol. 1043 of LNCS, Springer, 1995, pp. 238–266.

[8] E. M. Clarke, O. Grumberg, K. Hamaguchi, Another look at LTL model checking, Formal Methods in System Design 10 (1) (1997) 47–71.

[9] A. Biere, C. Artho, V. Schuppan, Liveness checking as safety checking, Electron. Notes Theor. Comput. Sci. 66 (2) (2002) 160–177. doi:10.1016/S1571-0661(04)80410-9.

[10] S. Graf, H. Saïdi, Construction of abstract state graphs with PVS, in: O. Grumberg (Ed.), Computer Aided Verification, 9th International Conference, CAV '97, Haifa, Israel, June 22-25, 1997, Proceedings, Vol. 1254 of Lecture Notes in Computer Science, Springer, 1997, pp. 72–83. doi:10.1007/3-540-63166-6_10.

[11] J. Daniel, A. Cimatti, A. Griggio, S. Tonetta, S. Mover, Infinite-state liveness-to-safety via implicit abstraction and well-founded relations, in: S. Chaudhuri, A. Farzan (Eds.), Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I, Vol. 9779 of Lecture Notes in Computer Science, Springer, 2016, pp. 271–291. doi:10.1007/978-3-319-41528-4_15.

[12] R. Alur, D. L. Dill, A theory of timed automata, Theor. Comput. Sci. 126 (2) (1994) 183–235. doi:10.1016/0304-3975(94)90010-8.

[13] A. Cimatti, A. Griggio, E. Magnago, M. Roveri, S. Tonetta, Extending nuxmv with timed transition systems and timed temporal properties, in: I. Dillig, S. Tasiran (Eds.), Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I, Vol. 11561 of Lecture Notes in Computer Science, Springer, 2019, pp. 376–386. doi:10.1007/978-3-030-25540-4_21.

[14] T. A. Henzinger, The theory of hybrid automata, in: Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996, IEEE Computer Society, 1996, pp. 278–292. doi:10.1109/LICS.1996.561342.
URL http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4265

[15] A. Asadi, K. Chatterjee, H. Fu, A. K. Goharshady, M. Mahdavi, Polynomial reachability witnesses via stellensätze, in: S. N. Freund, E. Yahav (Eds.), PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 20211, ACM, 2021, pp. 772–787. doi:10.1145/3453483.3454076.

[16] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, Y. Zhu, Bounded model checking, Adv. Comput. 58 (2003) 117–148. doi:10.1016/S0065-2458(03)58003-2.

[17] J. Leike, M. Heizmann, Ranking templates for linear loops, Log. Methods Comput. Sci. 11 (1). doi:10.2168/LMCS-11(1:16)2015.

[18] R. Bagnara, F. Mesnard, A. Pescetti, E. Zaffanella, A new look at the automatic synthesis of linear ranking functions, Inf. Comput. 215 (2012) 47–67. doi:10.1016/j.ic.2012.03.003.

[19] A. R. Bradley, Z. Manna, H. B. Sipma, Linear ranking with reachability, in: K. Etessami, S. K. Rajamani (Eds.), Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings, Vol. 3576 of Lecture Notes in Computer Science, Springer, 2005, pp. 491–504. doi:10.1007/11513988_48.

[20] D. Déharbe, P. Fontaine, D. L. Berre, B. Mazure, Computing prime implicants, in: Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013, IEEE, 2013, pp. 46–52.
URL http://ieeexplore.ieee.org/document/6679390/

[21] A. Previti, A. Ignatiev, A. Morgado, J. Marques-Silva, Prime compilation of non-clausal formulae, in: Q. Yang, M. J. Wooldridge (Eds.), Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, AAAI Press, 2015, pp. 1980–1988.
URL http://ijcai.org/Abstract/15/281

[22] A. Gupta, T. A. Henzinger, R. Majumdar, A. Rybalchenko, R. Xu, Proving non-termination, in: G. C. Necula, P. Wadler (Eds.), Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008, ACM, 2008, pp. 147–158. doi:10.1145/1328438.1328459.
URL http://dl.acm.org/citation.cfm?id=1328438

[23] H. Y. Chen, B. Cook, C. Fuhs, K. Nimkar, P. W. O'Hearn, Proving nontermination via safety, in: E. Ábrahám, K. Havelund (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings, Vol. 8413 of Lecture Notes in Computer Science, Springer, 2014, pp. 156–171. doi:10.1007/978-3-642-54862-8_11.

[24] D. Larraz, K. Nimkar, A. Oliveras, E. Rodríguez-Carbonell, A. Rubio, Proving non-termination using max-smt, in: A. Biere, R. Bloem (Eds.), Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings, Vol. 8559 of Lecture Notes in Computer Science, Springer, 2014, pp. 779–796. doi:10.1007/978-3-319-08867-9_52.

[25] F. Frohn, J. Giesl, Termination of triangular integer loops is decidable, in: I. Dillig, S. Tasiran (Eds.), Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part II, Vol. 11562 of Lecture Notes in Computer Science, Springer, 2019, pp. 426–444. doi:10.1007/978-3-030-25543-5_24.

[26] M. Hosseini, J. Ouaknine, J. Worrell, Termination of linear loops over the integers, in: C. Baier, I. Chatzigiannakis, P. Flocchini, S. Leonardi (Eds.), 46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece, Vol. 132 of LIPIcs,

Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 118:1–118:13. doi:10.4230/LIPIcs.ICALP.2019.118.

[27] J. Leike, M. Heizmann, Geometric nontermination arguments, in: D. Beyer, M. Huisman (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II, Vol. 10806 of Lecture Notes in Computer Science, Springer, 2018, pp. 266–283. doi:10.1007/978-3-319-89963-3_16.

[28] B. Cook, E. Koskinen, M. Y. Vardi, Temporal property verification as a program analysis task - extended version, Formal Methods Syst. Des. 41 (1) (2012) 66–82. doi:10.1007/s10703-012-0153-5.

[29] B. Cook, H. Khlaaf, N. Piterman, Verifying increasingly expressive temporal logics for infinite-state systems, J. ACM 64 (2) (2017) 15:1–15:39. doi:10.1145/3060257.

[30] Y. Kesten, A. Pnueli, A compositional approach to ctl* verification, Theor. Comput. Sci. 331 (2-3) (2005) 397–428. doi:10.1016/j.tcs.2004.09.023.

[31] Y. Kesten, A. Pnueli, L. Raviv, Algorithmic verification of linear temporal logic specifications, in: K. G. Larsen, S. Skyum, G. Winskel (Eds.), Automata, Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark, July 13-17, 1998, Proceedings, Vol. 1443 of Lecture Notes in Computer Science, Springer, 1998, pp. 1–16. doi:10.1007/BFb0055036.

[32] Y. Kesten, A. Pnueli, L. Raviv, E. Shahar, Model checking with strong fairness, Formal Methods Syst. Des. 28 (1) (2006) 57–84. doi:10.1007/s10703-006-4342-y.

[33] T. A. Beyene, C. Popeea, A. Rybalchenko, Solving existentially quantified horn clauses, in: N. Sharygina, H. Veith (Eds.), Computer Aided Verifica-

tion - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings, Vol. 8044 of Lecture Notes in Computer Science, Springer, 2013, pp. 869–882. doi:10.1007/978-3-642-39799-8_61.

[34] G. Behrmann, A. David, K. G. Larsen, A tutorial on UPPAAL, in: M. Bernardo, F. Corradini (Eds.), Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004, no. 3185 in LNCS, Springer–Verlag, 2004, pp. 200–236.

[35] R. Kindermann, T. A. Junttila, I. Niemelä, Beyond lassos: Complete smt-based bounded model checking for timed automata, in: H. Giese, G. Rosu (Eds.), Formal Techniques for Distributed Systems - Joint 14th IFIP WG 6.1 International Conference, FMOODS 2012 and 32nd IFIP WG 6.1 International Conference, FORTE 2012, Stockholm, Sweden, June 13-16, 2012. Proceedings, Vol. 7273 of Lecture Notes in Computer Science, Springer, 2012, pp. 84–100. doi:10.1007/978-3-642-30793-5_6.

[36] R. Alur, Formal verification of hybrid systems, in: S. Chakraborty, A. Jerraya, S. K. Baruah, S. Fischmeister (Eds.), Proceedings of the 11th International Conference on Embedded Software, EMSOFT 2011, part of the Seventh Embedded Systems Week, ESWeek 2011, Taipei, Taiwan, October 9-14, 2011, ACM, 2011, pp. 273–278. doi:10.1145/2038642.2038685.

[37] D. Bresolin, Hyltl: a temporal logic for model checking hybrid systems, in: L. Bortolussi, M. L. Bujorianu, G. Pola (Eds.), Proceedings Third International Workshop on Hybrid Autonomous Systems, HAS 2013, Rome, Italy, 17th March 2013, Vol. 124 of EPTCS, 2013, pp. 73–84. doi:10.4204/EPTCS.124.8.

[38] A. Cimatti, A. Griggio, S. Mover, S. Tonetta, Verifying LTL properties of hybrid systems with k-liveness, in: A. Biere, R. Bloem (Eds.), Computer Aided Verification - 26th International Conference, CAV 2014, Held as

Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings, Vol. 8559 of Lecture Notes in Computer Science, Springer, 2014, pp. 424–440. doi:10.1007/978-3-319-08867-9_28.

[39] A. Podelski, S. Wagner, Region stability proofs for hybrid systems, in: J. Raskin, P. S. Thiagarajan (Eds.), Formal Modeling and Analysis of Timed Systems, 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007, Proceedings, Vol. 4763 of Lecture Notes in Computer Science, Springer, 2007, pp. 320–335. doi:10.1007/978-3-540-75454-1_23.

[40] T. Nghiem, S. Sankaranarayanan, G. E. Fainekos, F. Ivancic, A. Gupta, G. J. Pappas, Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems, in: K. H. Johansson, W. Yi (Eds.), Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2010, Stockholm, Sweden, April 12-15, 2010, ACM, 2010, pp. 211–220. doi:10.1145/1755952.1755983.

[41] W. Damm, G. Pinto, S. Ratschan, Guaranteed termination in the verification of ltl properties of non-linear robust discrete time hybrid systems, Int. J. Found. Comput. Sci. 18 (1) (2007) 63–86. doi:10.1142/S0129054107004577.

[42] E. Plaku, L. E. Kavraki, M. Y. Vardi, Falsification of LTL safety properties in hybrid systems, Int. J. Softw. Tools Technol. Transf. 15 (4) (2013) 305–320. doi:10.1007/s10009-012-0233-2.

[43] S. Grebenshchikov, N. P. Lopes, C. Popeea, A. Rybalchenko, Synthesizing software verifiers from proof rules, in: J. Vitek, H. Lin, F. Tip (Eds.), ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '12, Beijing, China - June 11 - 16, 2012, ACM, 2012, pp. 405–416. doi:10.1145/2254064.2254112.

[44] T. A. Beyene, S. Chaudhuri, C. Popeea, A. Rybalchenko, A constraint-based approach to solving games on infinite graphs, in: S. Jagannathan,

P. Sewell (Eds.), The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014, ACM, 2014, pp. 221–234. doi:10.1145/2535838.2535860.

[45] A. Gurfinkel, N. Bjørner, The science, art, and magic of constrained horn clauses, in: 21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2019, Timisoara, Romania, September 4-7, 2019, IEEE, 2019, pp. 6–10. doi:10.1109/SYNASC49474.2019.00010.

[46] A. Cimatti, A. Griggio, B. J. Schaafsma, R. Sebastiani, The mathsat5 SMT solver, in: N. Piterman, S. A. Smolka (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings, Vol. 7795 of Lecture Notes in Computer Science, Springer, 2013, pp. 93–107. doi:10.1007/978-3-642-36742-7_7.

[47] L. M. de Moura, N. Bjørner, Z3: an efficient SMT solver, in: C. R. Ramakrishnan, J. Rehof (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings, Vol. 4963 of Lecture Notes in Computer Science, Springer, 2008, pp. 337–340. doi:10.1007/978-3-540-78800-3_24.

[48] M. Gario, A. Micheli, Pysmt: a solver-agnostic library for fast prototyping of smt-based algorithms, in: SMT Workshop 2015, 2015.

[49] B. Dutertre, Solving exists/forall problems with yices, in: Workshop on satisfiability modulo theories, 2015.

[50] T. S. Motzkin, Two consequences of the transposition theorem on linear inequalities, Econometrica (pre-1986) 19 (2) (1951) 184.

[51] J. Giesl, A. Rubio, C. Sternagel, J. Waldmann, A. Yamada, The termination and complexity competition, in: D. Beyer, M. Huisman, F. Kordon, B. Steffen (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 25 Years of TACAS: TOOLympics, Held as Part of ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part III, Vol. 11429 of Lecture Notes in Computer Science, Springer, 2019, pp. 156–166. doi:10.1007/978-3-030-17502-3_10.

[52] R. Farkas, G. Bergmann, Towards reliable benchmarks of timed automata, in: B. Pataki (Ed.), Proceedings of the 25th PhD Mini-Symposium, Budapest University of Technology and Economics, Department of Measurement and Information Systems, 2018, pp. 20–23.

[53] G. Frehse, M. Althoff (Eds.), ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systemsi, part of CPS-IoT Week 2019, Montreal, QC, Canada, April 15, 2019, Vol. 61 of EPiC Series in Computing, EasyChair, 2019.

[54] J. Giesl, M. Brockschmidt, F. Emmes, F. Frohn, C. Fuhs, C. Otto, M. Plücker, P. Schneider-Kamp, T. Ströder, S. Swiderski, R. Thiemann, Proving termination of programs automatically with aprove, in: S. Demri, D. Kapur, C. Weidenbach (Eds.), Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings, Vol. 8562 of Lecture Notes in Computer Science, Springer, 2014, pp. 184–191. doi:10.1007/978-3-319-08587-6_13.

[55] J. Havlíček, Untimed ltl model checking of timed automata, Ph.D. thesis, Master's thesis. Masaryk University, Faculty of Informatics, 2013. (2013). URL https://theses.cz/id/t1s8vb/

[56] J. Doménech, S. Genaim, irankfinder, WST 18 (2018) 83.

[57] R. Kindermann, T. A. Junttila, I. Niemelä, Bounded model checking of an MITL fragment for timed automata, in: J. Carmona, M. T. Lazarescu,

M. Pietkiewicz-Koutny (Eds.), 13th International Conference on Application of Concurrency to System Design, ACSD 2013, Barcelona, Spain, 8-10 July, 2013, IEEE Computer Society, 2013, pp. 216–225. doi:10.1109/ACSD.2013.25.

[58] M. Brockschmidt, B. Cook, S. Ishtiaq, H. Khlaaf, N. Piterman, T2: temporal property verification, in: M. Chechik, J. Raskin (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings, Vol. 9636 of Lecture Notes in Computer Science, Springer, 2016, pp. 387–393. doi:10.1007/978-3-662-49674-9_22.

[59] M. Heizmann, J. Christ, D. Dietsch, E. Ermis, J. Hoenicke, M. Lindenmann, A. Nutz, C. Schilling, A. Podelski, Ultimate automizer with smtinterpol - (competition contribution), in: N. Piterman, S. A. Smolka (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings, Vol. 7795 of Lecture Notes in Computer Science, Springer, 2013, pp. 641–643. doi:10.1007/978-3-642-36742-7_53.

[60] A. David, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, Uppaal smc tutorial, International Journal on Software Tools for Technology Transfer 17 (4) (2015) 397–415. doi:10.1007/s10009-014-0361-y.

[61] R. Alur, T. Feder, T. Henzinger, The Benefits of Relaxing Punctuality, J. ACM 43 (1) (1996) 116–146.

[62] A. Rabinovich, On the Decidability of Continuous Time Specification Formalisms, J. Log. Comput. 8 (5) (1998) 669–678.

Appendix A. Encoding of funnel-loop search in E-CHC

We described the funnel-loop synthesis problem using exist-forall quantified First-Order Logic formulae and defined an ad-hoc procedure to address this problem. In the literature it is possible to find formalism meant to decouple the definition of the verification problem and the actual solving algorithm: they separate the proof methodology from the procedures used to address the problem. One such formalism is *Constrained Horn clauses* (CHCs) [43, 44, 45], for which off-the-shelf solvers have been developed. In the following, we consider an extension of CHCs, namely *existentially-quantified constrained Horn-like clauses* (E-CHCs) [33]. E-CHCs are expressive enough to represent the funnel-loop synthesis problem, hence allow an alternative representation of our synthesis problem. However, possibly due to the complexity of solving such problems in general, there is a lack of tools capable of identifying solution for E-CHCs. Therefore, the encoding in E-CHCs has no impact on the theoretical and experimental results we present in this work. We believe that representing the problem in a common framework to describe verification tasks could provide a better perspective on how the approach and techniques we propose in this work fit into the broader context of formal verification.

For the syntax and semantics of E-CHCs we refer to [43]. In the following we present a sound and complete encoding for the search problem of a funnel-loop of length one in E-CHCs. It is possible to define a similar E-CHC encoding that also considers a set of user-defined E-components as hints, similarly to the procedure described in Sec. 7. However, such encoding is rather complex and does not provide any additional contribution to our discussion since we were not able to obtain any tool capable of identifying solutions for E-CHCs.

Let $M \doteq \langle V, I^M, T^M, F^M \rangle$ be a fair transition system and let $R(c, V)$, $T(V, V')$ and $Rank(V, V')$ be query symbols, where c is a fresh Boolean symbol ($c \notin V$). A solution to the E-CHC problem below is an intepretation for R, T and $Rank$ satisfying all its formulae. R represents the source region, and $c \wedge R(c, V)$ underapproximates the fair states. Th. 7 shows that any solution to the E-CHC below corresponds to a funnel-loop and Th. 8 shows that if M admits a funnel-loop then there exists an interpretation for the query symbols satisfying the following E-CHC. Therefore, the encoding is sound (Th. 7) and relatively complete (Th. 8). While it is possible to represent in E-CHC the search of a funnel-loop of arbitrary length n, Th. 2 ensures that looking for funnel-loops of length one is sufficient.

$$\top \;\rightarrow\; \exists c, V : R(c, V) \wedge I^M(V) \tag{A.1}$$

$$T(V, V') \;\rightarrow\; \exists c : R(c, V') \tag{A.2}$$

$$R(c, V) \wedge T(V, V') \;\rightarrow\; T^M(V, V') \tag{A.3}$$

$$R(c, V) \;\rightarrow\; \exists V' : T(V, V') \tag{A.4}$$

$$c \wedge R(c, V) \;\rightarrow\; F^M(V) \tag{A.5}$$

$$\neg c \wedge R(c, V) \wedge T(V, V') \;\rightarrow\; \mathit{Rank}(V, V') \tag{A.6}$$

$$\mathit{dwf}(\mathit{Rank}) \tag{A.7}$$

Let $\mathit{Cex} \doteq \langle V, \exists c : R(c, V), T(V, V'), \top\rangle$ be the transition system associated with an interpretation of the query symbols of the E-CHC above. Eq. (A.1) requires the existence of some initial state of M in R: the set of initial states of Cex is not empty. Eq. (A.2) ensures that T can only reach states in R, and Eq. (A.3) guarantees that in such region T is an underapproximation of T^M: Cex is simulated by M, hence a path of Cex is also a path in M. Eq. (A.4) requires T to be left-total with respect to R: Cex cannot reach a deadlock. Eq. (A.5) requires $R(\bot, V)$ to be a subset of the fair states of M. Eq. (A.6) requires the relation $T(V, V')$ describing pairs of current and next states such that the first one is in $R(\bot, V)$ to underapproximate some well-founded relation Rank. The well-foundedness of Rank ensures that there is no infinite chain of states in $R(\bot, V)$, hence it must eventually reach a state in $R(\top, V)$, hence by Eq. (A.5) must eventually reach a fair state.

Theorem 7. *Given a fair transition system $M \doteq \langle V, I^M, T^M, F^M\rangle$ and an interpretation for the queries R, T and Rank satisfying all Eqs.* (A.1)–(A.7)*. Then* *there exist a funnel-loop for M.*

The proof of Th. 7 is reported in Appendix B.5.

Theorem 8. *Let $floop$ be a funnel-loop of length one for a transition system $M \doteq \langle V, I^M, T^M, F^M\rangle$. Then, there exists an intepretation for the query symbols R, T and $Rank$ satisfying all Eqs.* (A.1)–(A.7).

The proof of Th. 8 is reported in Appendix B.6.

Appendix B. Theorems and proofs

Appendix B.1. Funnel-loop disjoint regions

In this section we show that for every funnel-loop there exist a corresponding one whose regions are pairwise disjoint and that admits the same paths. Let $floop \doteq [fnl_i]_{i=0}^{n-1}$ be a funnel-loop of length n over symbols V. We define a corresponding funnel-loop $\widehat{floop} \doteq [\widehat{fnl}_i]_{i=0}^{n-1}$ over symbols $\widehat{V} \doteq V \cup \{l\}$ that admits the same set of paths projected over the symbols V and whose regions are pairwise disjoint. l is a fresh symbol ($l \notin V$) we use to keep track of the index of the current region. More formally we have the following:

- $\widehat{V} \doteq V \cup \{l\}$, where $l \notin V$ is a fresh symbol whose domain are the integers from 0 to $n-1$.
- $\widehat{S}_i \doteq S_i \wedge l = i$.
- $\widehat{D}_i \doteq D_i \wedge l = (i+1)\%n$.
- $\widehat{\mathrm{RF}}_i \doteq \mathrm{RF}_i$.
- $\widehat{T}_i \doteq T_i \wedge (\mathbf{0} < \mathrm{RF}_i \wedge l' = l) \vee (\mathrm{RF}_i = \mathbf{0} \wedge l' = (l+1)\%n)$

Theorem 9. *Let $floop$ be a funnel-loop. Then, all $[\widehat{fnl}_i]_{i=0}^{n-1}$ satisfy the hypotheses of Def. 1 and $\widehat{floop}$ satisfies the hypotheses of Def. 2.*

Proof. We first show that each $\widehat{fnl}_i$ in $[\widehat{fnl}_i]_{i=0}^{n-1}$ is a funnel and then show that they are correctly concatenated in $\widehat{floop}$ hence it is a funnel-loop.

F.1 By definition $\widehat{T_i} \doteq T_i \wedge (\mathbf{0} < \textsc{Rf}_i \wedge l' = l) \vee (\textsc{Rf}_i = \mathbf{0} \wedge l' = (l+1)\%n)$. In each state either $\textsc{Rf}_i = \mathbf{0}$ holds or $\mathbf{0} < \textsc{Rf}_i$ does. Therefore, in the first case $\widehat{T_i}$ admits a successor in such that $l' = (l+1)\%n$, in the second case it admits a successor in which $l' = l$. Since Hyp. F.1 holds for fnl_i, its transition relation $T_i(V, V')$ is left-total. Therefore, also $\widehat{T_i}$ is left-total and Hyp. F.1 holds for each $\widehat{fnl_i}$ in $\widehat{floop}$.

F.2 By definition $\widehat{T_i} \doteq T_i \wedge (\mathbf{0} < \textsc{Rf}_i \wedge l' = l) \vee (\textsc{Rf}_i = \mathbf{0} \wedge l' = (l+1)\%n)$. Therefore, every pair of states $\langle \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{v}}' \rangle \in \widehat{T_i}$ such that $\widehat{\boldsymbol{v}} \models \widehat{S_i}$ must be such that they assign the same value i to l. Let $\boldsymbol{v}$ and $\boldsymbol{v}'$ be the projection of $\widehat{\boldsymbol{v}}$ $\widehat{\boldsymbol{v}}'$ to the symbols V, Then, $\langle \boldsymbol{v}, \boldsymbol{v}' \rangle \in T_i$ and $\boldsymbol{v} \models S_i$. Since, Hyp. F.2 holds for fnl_i, then $\boldsymbol{v}' \models S_i'$ also holds. $\boldsymbol{v}' \models S_i'$ and the fact that $\widehat{\boldsymbol{v}}'$ assigns l to i implies that $\widehat{\boldsymbol{v}}' \models S'$. Therefore, Hyp. F.2 holds for $\widehat{fnl_i}$.

F.3 By applying the same reasoning as above, for every such step in $\widehat{fnl_i}$ we obtain a corresponding step in fnl_i by projecting the assignments over the symbols in V. Hyp. F.3 holds for fnl_i hence those assignments must decrease the value of the ranking function $\textsc{Rf}_i$. Therefore, since $\textsc{Rf}_i$ does not depend on l its value must decrease also in all such steps of $\widehat{fnl_i}$ and Hyp. F.3 must hold.

F.4 By applying the same reasoning as the previous two cases, for every such step in $\widehat{fnl_i}$ we obtain a corresponding step in fnl_i by projecting the assignments over the symbols in V. Hyp. F.4 holds for fnl_i hence the second projected state must be in D_i. By definition of $\widehat{T}$, the second state must assign to l the index of the next region. Such an assignment agrees with the assignment required by $\widehat{D}_i$, therefore Hyp. F.4 holds for $\widehat{fnl_i}$.

We now show that $\widehat{floop}$ is a funnel-loop.

FL.1, FL.2 Each $\widehat{D}_i$ requires the same assignment to l as $\widehat{S}_{(i+1)\%n}$. Therefore, since hypotheses FL.1 and FL.2 holds for $floop$, they must also hold for $\widehat{floop}$.

$\square$

Theorem 10. *The languages of $floop$ and $\widehat{floop}$ admit the same set of paths projected over the symbols V.*

Proof. We show that $\widehat{floop}$ admits all paths of $floop$ and vice-versa by induction of their funnels and the length of the path.

- Assume $floop$ admits a path starting from some state $\boldsymbol{v}$. Then by definition $\boldsymbol{v} \models S_i$ for some i. Let $\widehat{\boldsymbol{v}}$ assign l to i and agree with $\boldsymbol{v}$ on the assignments of all symbols in V. Then $\widehat{\boldsymbol{v}} \models \widehat{S}_i$ and $\widehat{\boldsymbol{v}}$ is an initial state for $\widehat{floop}$.

  Viceversa, assume $\widehat{floop}$ admits a path starting from some state $\widehat{\boldsymbol{v}}$. Then by definition $\widehat{\boldsymbol{v}} \models \widehat{S}_i$ for some i. Let $\boldsymbol{v}$ be its projection over the symbols V, then $\boldsymbol{v} \models S_i$ and is an initial state for $floop$.

- Let π be a path of $floop$ ending in state $\boldsymbol{v}$ and $\widehat{\pi}$ be the correspoding path of $\widehat{floop}$ ending in $\widehat{\boldsymbol{v}}$. Let S_i be the region of $floop$ such that $\boldsymbol{v} \models S_i$ and let $\widehat{S}_i$ be its corresponding region in $\widehat{floop}$.

  Assume $floop$ admits a successor state $\boldsymbol{v}'$ of $\boldsymbol{v}$. Then either $\boldsymbol{v}' \models S'_i$ or $\boldsymbol{v}' \models S'_{(i+1)\%n}$. Let $\widehat{\boldsymbol{v}}'$ be the assignment that extends $\boldsymbol{v}'$ with $l' = i$ in the first case and $l' = (i+1)\%n$ otherwise. $\widehat{\boldsymbol{v}}'$ is a successor of $\widehat{\boldsymbol{v}}$ corresponding to $\boldsymbol{v}'$ such that π extended with $\boldsymbol{v}'$ corresponds to $\widehat{\pi}$ extended with $\widehat{\boldsymbol{v}}'$.

  Viceversa, assume $\widehat{floop}$ admits a successor state $\widehat{\boldsymbol{v}}'$ of $\widehat{\boldsymbol{v}}$. Let $\boldsymbol{v}'$ be the restriction of $\widehat{\boldsymbol{v}}'$ to the symbols in V. Then, $\boldsymbol{v}'$ is a successor for $\boldsymbol{v}$ such that $\widehat{\pi}$ extended with $\widehat{\boldsymbol{v}}'$ corresponds to π extended with $\boldsymbol{v}'$.

$\square$

Appendix B.2. E-components disjoint regions

In this section we show that for every E-component there exist a corresponding one whose regions are pairwise disjoint and that admits the same paths. Given an E-component $H \doteq \langle V, I(V), T(V, V') \rangle$ of length m over regions $\mathcal{R}$, assumptions $\mathcal{A}$, ranking functions $\mathcal{W}$ and responsible for $V_r \subseteq V$, we define a corresponding E-component $\widehat{H} \doteq \langle \hat{V}, \hat{I}(\hat{V}), \hat{T}(\hat{V}, \hat{V}') \rangle$ over regions $\hat{\mathcal{R}}$, assumptions $\mathcal{A}$, ranking functions $\mathcal{W}$ and responsible for $\hat{V}_r$ whose regions and pairwise disjoint. $\widehat{H}$, with respect to H, has an additional symbol l used to keep track of the index of the current region and each region is strengthened by requiring the correct assignment for such symbol, while the sets of assumptions and ranking functions remain the same. More formally we have the following:

- $\hat{V} \doteq V \cup \{l\}$, where $l \notin V$ is a fresh symbol whose domain are the integers from 0 to $m-1$.
- $\hat{V}_r \doteq V_r \cup \{l\}$.
- $\hat{\mathcal{R}} \doteq \{R_j \wedge l = j \mid R_j \in \mathcal{R}\}$
- $\hat{I}(\hat{V}) \doteq I(V) \wedge \bigwedge_{j=0}^{m-1} l = j \rightarrow R_j(V) \wedge A_j(V)$.
- $\hat{T}(\hat{V}, \hat{V}') \doteq T(V, V') \wedge \bigwedge_{j=0}^{m-1} l' = j \rightarrow R_j(V')$

Notice that by construction the $\hat{\mathcal{R}}$ are pairwise disjoint. We now show that H and $\widehat{H}$ admit the same paths with respect to the assignments over the common symbols V and that $\widehat{H}$ is in fact an E-component.

Theorem 11. *If H satisfies all hypotheses of Def. 3 then also $\widehat{H}$ does.*

Proof.

I By hypothesis $I \rightarrow \bigvee_{j=0}^{m-1} R_j \wedge A_j$ holds and we need to show that

$$(I \wedge \bigwedge_{j=0}^{m-1} l = j \rightarrow (R_j \wedge A_j)) \rightarrow \bigvee_{j=0}^{m-1} R_j \wedge A_j \wedge l = j$$

also holds. By hypothesis, for every state $\boldsymbol{v}$ in I there must exist some j_0 such that $\boldsymbol{v} \models R_{j_0} \wedge A_{j_0}$. By definition of l, $\bigvee_{j=0}^{m-1} l = j$ is valid, hence the left-hand-side of the implication $l = j \rightarrow (R_j \wedge A_j)$ cannot be always false. Consider an assignment $\widehat{\boldsymbol{v}}$ over $\hat{V}$ and let j_0 such that $\widehat{\boldsymbol{v}} \models l = j_0$. Then, if $\widehat{\boldsymbol{v}} \not\models R_{j_0} \wedge A_{j_0}$ our objective formula holds. Otherwise, $\widehat{\boldsymbol{v}} \models R_{j_0} \wedge A_{j_0} \wedge l = j_0$, hence $\widehat{\boldsymbol{v}} \models \bigvee_{j=0}^{m-1} R_j \wedge A_j \wedge l = j$.

II, **III**, **IV** If $\widehat{H}$ admits a transition between two restricted regions $\hat{R}_{j_0} \wedge A_{j_0}$ and $\hat{R}_{j_1} \wedge A_{j_1}$ of one of the 3 kinds then, by construction of $\hat{T}$, H must admit a transition of the same kind between its restricted regions $R_{j_0} \wedge A_{j_0}$ and $R_{j_1} \wedge A_{j_1}$. Let t be the kind of the transition. All three hypotheses hold for H, hence every state in $R_{j_0} \wedge A_{j_0}$ admits at least one successor in R_{j_1} via a t-transition, provided A_{j_1} holds. For every state $\hat{\boldsymbol{v}}$ in $\hat{R}_{j_0} \wedge A_{j_0}$, let $\boldsymbol{v}$ be its restriction to the symbols in V. $\boldsymbol{v}$ is in $R_{j_0} \wedge A_{j_0}$ and it admits a successor $\boldsymbol{v}'$ via a t-transition. Then, $\hat{\boldsymbol{v}}'$ defined by extending $\boldsymbol{v}'$ with $l' = j_1$, is a t-successor for $\hat{\boldsymbol{v}}$ in $\widehat{H}$.

□

Theorem 12. *The languages of H and $\hat{H}$ admit the same set of paths projected over the symbols V.*

Proof. We prove the statement by induction on the length of the path. We first show that there is a one-to-one correspondence between the initial states and then show that a one-to-one correspondence exists also between the transitions.

- Every initial state $\boldsymbol{v}_0$ of H must be such that $I(\boldsymbol{v}_0)$ is true and, by Hyp. **I** there exists $0 \leq j_0 < m$ such that $R_{j_0}(\boldsymbol{v}_0) \wedge A_{j_0}(\boldsymbol{v}_0)$ also holds. We define an assignment $\hat{\boldsymbol{v}}_0$ over $\hat{V}$ by extending $\boldsymbol{v}_0$ with the assignment $l = j_0$. By construction, $\hat{I}(\hat{\boldsymbol{v}}_0)$ and $\hat{R}_{j_0}(\hat{\boldsymbol{v}}_0) \wedge A_{j_0}(\hat{\boldsymbol{v}}_0)$ hold, hence $\hat{\boldsymbol{v}}_0$ is an initial state for some path in $\mathcal{L}(\widehat{H})$.

  Viceversa, given an initial state $\hat{\boldsymbol{v}}_0$ of $\widehat{H}$, we define $\boldsymbol{v}_0$ by restricting $\hat{\boldsymbol{v}}_0$ to the assignments of the symbols in V. By construction, $I(\boldsymbol{v}_0)$ is true and there exists $0 \leq j_0 < m$ such that $R_{j_0}(\boldsymbol{v}_0) \wedge A_{j_0}(\boldsymbol{v}_0)$ holds. Therefore, $\boldsymbol{v}_0$ is an initial state for H.

- Consider a transition of H from assignment $\boldsymbol{v}$ to $\boldsymbol{v}'$: $\boldsymbol{v}, \boldsymbol{v}' \models R_j \wedge A_j \wedge T \wedge R_{j'} \wedge A_{j'}$ for some $0 \leq j < m$ and $0 \leq j' < m$. By inductive hypothesis there is an assignment $\widehat{\boldsymbol{v}}$ for the symbols $\widehat{V}$ corresponding to $\boldsymbol{v}$. We show that $\widehat{H}$ admits a successor $\widehat{\boldsymbol{v}}'$ for $\widehat{\boldsymbol{v}}$ that corresponds to $\boldsymbol{v}'$. By hypothesis,

$\boldsymbol{v}' \models R_{j'} \wedge A_{j'}$. We define $\widehat{\boldsymbol{v}}'$ by extending the assignment $\boldsymbol{v}'$ with $l = j'$. Then, $\widehat{\boldsymbol{v}}'$ corresponds to $\boldsymbol{v}'$ and $\widehat{\boldsymbol{v}}, \widehat{\boldsymbol{v}}' \models \widehat{R}_j \wedge A_j \wedge \widehat{T} \wedge \widehat{R}_{j'} \wedge \widehat{A}_{j'}$.

Viceversa, consider now a transition of $\widehat{H}$ from assignment $\widehat{\boldsymbol{v}}$ to $\widehat{\boldsymbol{v}}'$ and an assignment $\boldsymbol{v} \doteq \widehat{\boldsymbol{v}}_{\downarrow V}$ for the symbols V corresponding to $\widehat{\boldsymbol{v}}$. By hypothesis, $\widehat{\boldsymbol{v}} \models \widehat{R}_j \wedge A_j$ and $\widehat{\boldsymbol{v}}' \models \widehat{R}_{j'} \wedge A_{j'}$ for some j and j'. By definition of $\widehat{R}_{j'}$ the following holds: $\widehat{\boldsymbol{v}}' \models R_{j'}$. $\boldsymbol{v}' \doteq \widehat{\boldsymbol{v}}'_{\downarrow V}$ is an assignment over the symbols V corresponding to $\widehat{\boldsymbol{v}}'$. Since $R_{j'}$ and $A_{j'}$ do not depend on l and $\widehat{\boldsymbol{v}}' \models R_{j'} \wedge A_{j'}$, then $\boldsymbol{v}' \models R_{j'} \wedge A_{j'}$. Hence, $\boldsymbol{v}, \boldsymbol{v}' \models R_j \wedge A_j \wedge T \wedge R_{j'} \wedge A_{j'}$. Therefore, $\boldsymbol{v}'$ is a successor for $\boldsymbol{v}$ in H corresponding to $\widehat{\boldsymbol{v}}'$.

□

Appendix B.3. Projection of E-components is closed

Theorem 4. *The projection $H^{\downarrow}$ over indexes idxs of an E-component H over regions $\mathcal{R}$, assumptions $\mathcal{A}$ and ranking functions $\mathcal{W}$ is an E-component.*

Proof. We prove that hypotheses **I**–**IV** hold for $H^{\downarrow}$.

I holds by construction since every state $\boldsymbol{v}$ such that $I^{\downarrow}(\boldsymbol{v})$ must also satisfy $\bigvee_{j \in idxs}(R_j(V) \wedge A_j(V))$ hence, by definition of $\mathcal{R}^{\downarrow}$ and $\mathcal{A}^{\downarrow}$, $\boldsymbol{v}$ is also in some restricted region of $H^{\downarrow}$.

II For any $j^{\downarrow} \in idxs$, the region $R^{\downarrow}_{j^{\downarrow}}$, assumption $A^{\downarrow}_{j^{\downarrow}}$ and ranking function $\text{RF}^{\downarrow}_{j^{\downarrow}}$ are in both $H^{\downarrow}$ and H. In all transitions such that $R_j \wedge \text{RF}'_j < \text{RF}_j \wedge R'_j$ holds for some $j \in idxs$, $T^{\downarrow}$ is equivalent to T. Therefore, since Hyp. **II** holds for H, it must also hold for $H^{\downarrow}$: if T admits a successor for every state in $R_j \wedge A_j$ such that $\text{RF}'_j < \text{RF}_j \wedge R'_j$ hold, then so does $T^{\downarrow}$.

III By construction of $T^{\downarrow}$ admits no stutter transition. Therefore, the left-hand-side of the entailment is false and Hyp. **III** holds.

IV For any $j^{\downarrow}, j^{\downarrow\prime} \in idxs$, if they do not denote consecutive regions in the sequence, $H^{\downarrow}$ does not admit any transition between them and Hyp. **IV** holds. Otherwise, $j^{\downarrow}$ and $j^{\downarrow\prime}$ are the consecutive indexes of the regions $R^{\downarrow}_{j^{\downarrow}}$,

$R^{\downarrow}_{j^{\downarrow\prime}}$, the assumptions $A^{\downarrow}_{j^{\downarrow}}$, $A^{\downarrow}_{j^{\downarrow\prime}}$ and ranking functions $\textsc{Rf}^{\downarrow}_{j^{\downarrow}}$. If H does not admit any progress transition between these regions, neither does $H^{\downarrow}$ and Hyp. **IV** holds. Otherwise if H admits at least one transition between these regions, the following holds:

$$\exists V, V' : R^{\downarrow}_{j^{\downarrow}} \wedge A^{\downarrow}_{j^{\downarrow}} \wedge \textsc{Rf}^{\downarrow}_{j^{\downarrow}} = \mathbf{0} \wedge T \wedge R^{\downarrow}_{j^{\downarrow\prime}} \wedge A^{\downarrow}_{j^{\downarrow\prime}}$$

Every such V and V' satisfies $T^{\downarrow}$, hence it is also a transition for $H^{\downarrow}$. Therefore, since Hyp. **IV** holds for H, for every state in $R^{\downarrow}_{j^{\downarrow}} \wedge A^{\downarrow}_{j^{\downarrow}} \wedge \textsc{Rf}^{\downarrow}_{j^{\downarrow}} = \mathbf{0}$ H admits a successor in $R^{\downarrow}_{j^{\downarrow\prime}} \wedge A^{\downarrow}_{j^{\downarrow\prime}}$. Every such transition is also admitted by $H^{\downarrow}$ and Hyp. **IV** holds for $H^{\downarrow}$.

□

Appendix B.4. Composition of E-components is closed

Theorem 5. *Given a set of E-components $\{H^i\}_{i=0}^{n}$, their composition $H^c \doteq \bigotimes_{i=0}^{n} H^i = \langle V, I^c, T^c \rangle$ is an E-component with respect to regions $\mathcal{R}^c$, assumptions $\mathcal{A}^c$ and ranking functions $\mathcal{W}^c$.*

Proof. We need to prove that hypotheses **I**–**IV** hold for H^c of length m^c. In the following we write $A^{i,\neq c}_{j_i}$ for $\bigwedge_{h \notin \{0,\ldots,n\}} A^{i,h}_{j_i}(V^h)$.

I requires us to prove that the initial states of H^c are a subset of the union of the regions. This holds trivially from the definition of I^c since every state in this set must satisfy $\bigvee_{j_c=0}^{m^c} R^c_{j_c} \wedge A^c_{j_c}$.

II requires us to prove the following

$$\begin{aligned}
&\forall j : 0 \leq j < m^c \rightarrow \\
&\quad \exists V, V' : (R^c_j \wedge A^c_j \wedge T^c \wedge \textsc{Rf}^{c\prime}_j < \textsc{Rf}^c_j \wedge R^{c\prime}_j \wedge A^{c\prime}_j) \quad \models \\
&\quad \forall V \exists V^{c\prime} \forall V^{\neq c\prime} : R^c_j \wedge A^c_j \wedge \mathbf{0} < \textsc{Rf}^c_j \wedge A^{c\prime}_j \rightarrow R^{c\prime}_j \wedge T^c \wedge \textsc{Rf}^{c\prime}_j < \textsc{Rf}^c_j
\end{aligned}$$

$H^c \doteq \bigotimes_{i=0}^{n} H^i$ hence, by definition of $\otimes$, R^c_j and A^c_j are the conjunction of some region and assumptions of $\{H^i\}_{i=0}^{n}$. Therefore, we can rewrite it as follows:

$$\forall \{j_i\}_{i=0}^{n} : \ (\bigwedge_{i=0}^{n} 0 \leq j_i < m^i) \rightarrow$$

$$\exists V, V' : (\bigwedge_{i=0}^{n} R^i_{j_i} \wedge (\bigwedge_{h=0, h\neq i}^{n} A^{i,h}_{j_i}) \wedge A^{i,\neq c}_{j_i} \wedge T^i) \wedge \mathit{compatible}_{\{H^i\}_{i=0}^{n}} \wedge$$

$$\mathit{indepRank}_{\{H^i\}_{i=0}^{n}} \wedge \mathrm{RF}^{c'}_j < \mathrm{RF}^{c}_j \wedge (\bigwedge_{i=0}^{n} {R^i_{j_i}}' \wedge (\bigwedge_{h=0, h\neq i}^{n} A^{i,h'}_{j_i}) \wedge A^{i,\neq c'}_{j_i}) \quad \models$$

$$\forall V \exists \{V^{i'}\}_{i=0}^{n} \forall V^{\neq c'} : ((\bigwedge_{i=0}^{n} R^i_{j_i} \wedge (\bigwedge_{h=0, h\neq i}^{n} A^{i,h}_{j_i}) \wedge A^{i,\neq c}_{j_i}) \wedge A^{c'}_j \wedge \mathbf{0} < \mathrm{RF}^c_j) \rightarrow$$

$$((\bigwedge_{i=0}^{n} {R^i_{j_i}}' \wedge (\bigwedge_{h=0, h\neq i}^{n} A^{i,h'}_{j_i}) \wedge T^i) \wedge \mathit{compatible}_{\{H^i\}_{i=0}^{n}} \wedge \mathit{indepRank}_{\{H^i\}_{i=0}^{n}} \wedge$$

$$\mathrm{RF}^{c'}_j < \mathrm{RF}^c_j)$$

For any $0 \leq i \leq n$ $A^i_j(V^{\neq c}) \wedge \bigwedge_{h=0,h\neq i}^{n} A^{i,h}_{j_i}(V^h)$ is equivalent to $A^i_j(V^{\neq i})$. Therefore, our objective formula can be rewritten as:

$$\forall \{j_i\}_{i=0}^{n} : \ (\bigwedge_{i=0}^{n} 0 \leq j_i < m^i) \rightarrow$$

$$\exists V, V' : (\bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i} \wedge T^i \wedge {R^i_{j_i}}' \wedge {A^i_{j_i}}') \wedge \mathit{compatible}_{\{H^i\}_{i=0}^{n}} \wedge$$

$$\mathit{indepRank}_{\{H^i\}_{i=0}^{n}} \wedge \mathrm{RF}^{c'}_j < \mathrm{RF}^c_j \quad \models$$

$$\forall V \exists \{V^{i'}\}_{i=0}^{n} \forall V^{\neq c'} : ((\bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i}) \wedge A^{c'}_j \wedge \mathbf{0} < \mathrm{RF}^c_j) \rightarrow ((\bigwedge_{i=0}^{n} T^i \wedge {R^i_{j_i}}' \wedge$$

$$\bigwedge_{h=0,h\neq i}^{n} A^{i,h'}_{j_i}) \wedge \mathit{compatible}_{\{H^i\}_{i=0}^{n}} \wedge \mathit{indepRank}_{\{H^i\}_{i=0}^{n}} \wedge \mathrm{RF}^{c'}_j < \mathrm{RF}^c_j)$$

If $\mathit{indepRank}_{\{H^i\}_{i=0}^{n}}$ [resp. $\mathit{compatible}_{\{H^i\}_{i=0}^{n}}$] does not hold in the left-hand-side of the entailment the formula is trivially true. By definition of $\mathit{indepRank}_{\{H^i\}_{i=0}^{n}}$ [resp. $\mathit{compatible}_{\{H^i\}_{i=0}^{n}}$], if it holds in the left-hand-side of the entailment it must also hold on the right-hand-side, since on both sides V and V' belong to the same regions. Therefore, $\mathit{compatible}_{\{H^i\}_{i=0}^{n}}$ must hold and when both sides of the implication on the right-hand-side of the entailment hold, $\bigwedge_{i=0}^{n} \bigwedge_{h=0,h\neq i}^{n} A^{i,h}_{j_i'}(V^{h'})$ must be true. We can further simplify our objective formula as follows:

$$\forall \{j_i\}_{i=0}^{n}: \ (\bigwedge_{i=0}^{n} 0 \leq j_i < m^i) \rightarrow \exists V, V' : (\bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i} \wedge T^i \wedge {R^i_{j_i}}' \wedge {A^i_{j_i}}') \wedge$$
$$\mathit{compatible}_{\{H^i\}_{i=0}^n} \wedge \mathit{indepRank}_{\{H^i\}_{i=0}^n} \wedge \mathrm{RF}^{c\prime}_j < \mathrm{RF}^c_j \quad \models \forall V \exists \{V^{i\prime}\}_{i=0}^n \forall V^{\neq c\prime} :$$
$$((\bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i}) \wedge A^{c\prime}_j \wedge \mathbf{0} < \mathrm{RF}^c_j) \rightarrow ((\bigwedge_{i=0}^{n} T^i \wedge {R^i_{j_i}}') \wedge \mathrm{RF}^{c\prime}_j < \mathrm{RF}^c_j)$$

If the left-hand-side of the entailment is false, then the formula is trivially true. Therefore, assume that there exists a transition performing a self-loop on the restricted region $R^c_j \wedge A^c_j$ with *independent ranks* in which the sum of the ranking function decreases. Under this assumption, we need to prove the following for any $j \doteq \langle j_0, \ldots, j_n \rangle$ satisfying the above:

$$\forall V \exists \{V^{i\prime}\}_{i=0}^n \forall V^{\neq c\prime} :$$
$$((\bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i}) \wedge A^{c\prime}_j \wedge \mathbf{0} < \mathrm{RF}^c_j(V)) \rightarrow ((\bigwedge_{i=0}^{n} T^i \wedge {R^i_{j_i}}') \wedge \mathrm{RF}^{c\prime}_j < \mathrm{RF}^c_j)$$

Since $\mathit{indepRank}_{\{H^i\}_{i=0}^n}$ holds for indexes $\langle j_0, \ldots, j_n \rangle$ we have:

$$\bigwedge_{i=0}^{n} (\forall V : (\bigwedge_{h=0}^{n} R^h_{j_h} \wedge A^h_{j_h}) \rightarrow \mathrm{RF}^i_{j_i} = \mathbf{0}) \vee$$
$$\exists V, V' : (\bigwedge_{h=0}^{n} R^h_{j_h} \wedge A^h_{j_h} \wedge T^h \wedge {R^h_{j_h}}' \wedge {A^h_{j_h}}') \wedge$$
$${\mathrm{RF}^i_{j_i}}' < \mathrm{RF}^i_{j_i} \wedge (\bigwedge_{k=0, k\neq h}^{n} {\mathrm{RF}^i_{j_i}}' = \mathrm{RF}^i_{j_i}) \wedge \mathit{compatible}_{\{H^i\}_{i=0}^n}$$

In addition, since there exists a transition in the restricted regions such that RF^c_j decreases, there must be some $0 \leq i_r \leq n$ such that $\exists V : (\bigwedge_{h=0}^n R^h_{j_h}(V) \wedge A^h_{j_h}(V^{\neq h})) \wedge \mathbf{0} < \mathrm{RF}^{i_r}_{j_{i_r}}(V)$. Then, there exist compatible transitions in which its ranking function decreases $\mathrm{RF}^{i_r}_{j_r}(V') < \mathrm{RF}^{i_r}_{j_r}(V)$, while all other ranking function remain constant $\bigwedge_{i=0, i\neq i_r}^n \mathrm{RF}^i_{j_i}(V') = \mathrm{RF}^i_{j_i}(V)$. Hyp. II holds for H^{i_r}:

$$\forall j_r: \ 0 \leq j_r < m^{i_r} \rightarrow \exists V, V' : (R^{i_r}_{j_r} \wedge A^{i_r}_{j_r} \wedge T^{i_r} \wedge {\mathrm{RF}^{i_r}_{j_r}}' < \mathrm{RF}^{i_r}_{j_r} \wedge {R^{i_r}_{j_r}}' \wedge {A^{i_r}_{j_r}}') \quad \models$$
$$\forall V \exists {V^{i_r}}' \forall {V^{\neq i_r}}' : R^{i_r}_{j_r} \wedge A^{i_r}_{j_r} \wedge \mathbf{0} < \mathrm{RF}^{i_r}_{j_r} \wedge {A^{i_r}_{j_r}}' \rightarrow {R^{i_r}_{j_r}}' \wedge T^{i_r} \wedge {\mathrm{RF}^{i_r}_{j_r}}' < \mathrm{RF}^{i_r}_{j_r}$$

and Hyp. III holds for all $\{H^i\}_{i=0,i\neq i_r}^n$:

$$\forall j_i : 0 \leq j_i < m^i \rightarrow \exists V, V' : (R^i_{j_i} \wedge A^i_{j_i} \wedge T^i \wedge \mathrm{RF}^{i\,\prime}_{j_i} = \mathrm{RF}^i_{j_i} \wedge R^{i\,\prime}_{j_i} \wedge A^{i\,\prime}_{j_i}) \quad \models$$
$$\forall V \exists V^{i\prime} \forall V^{\neq i\prime} : R^i_{j_i} \wedge A^i_{j_i} \wedge A^{i\,\prime}_{j_i} \rightarrow R^{i\,\prime}_{j_i} \wedge T^i \wedge \mathrm{RF}^{i\,\prime}_{j_i} = \mathrm{RF}^i_{j_i}$$

If there is no transition in the intersection of the restricted regions such that RF^c_j decreases or they are not compatible, the objective formula trivially holds because the left-hand-side of the entailment is false. Then, the conjunction of the hypotheses for the $\{H^i\}_{i=0}^n$ implies:

$$\forall \{j_i\}_{i=0}^n \; : \; (\bigwedge_{i=0}^n 0 \leq j_i < m^i) \rightarrow$$
$$\exists V, V' : (\bigwedge_{i=0}^n (R^i_{j_i} \wedge A^i_{j_i} \wedge T^i \wedge R^{i\,\prime}_{j_i} \wedge A^{i\,\prime}_{j_i}) \wedge \mathrm{RF}^{i_r\,\prime}_{j_r} < \mathrm{RF}^{i_r}_{j_r} \wedge \bigwedge_{i=0,i\neq r}^n \mathrm{RF}^{i\,\prime}_{j_i} = \mathrm{RF}^i_{j_i}) \quad \models$$
$$\forall V \exists V^{i_r\prime} \forall V^{\neq i_r\prime} : R^{i_r}_{j_r} \wedge A^{i_r}_{j_r} \wedge \mathbf{0} < \mathrm{RF}^{i_r}_{j_r} \wedge A^{i_r\,\prime}_{j_r} \rightarrow R^{i_r\,\prime}_{j_r} \wedge T^{i_r} \wedge \mathrm{RF}^{i_r\,\prime}_{j_r} < \mathrm{RF}^{i_r}_{j_r} \wedge$$
$$\bigwedge_{i=0,i\neq r}^n \forall V \exists V^{i\prime} \forall V^{\neq i\prime} : R^i_{j_i} \wedge A^i_{j_i} \wedge A^{i\,\prime}_{j_i} \rightarrow R^{i\,\prime}_{j_i} \wedge T^i \wedge \mathrm{RF}^{i\,\prime}_{j_i} = \mathrm{RF}^i_{j_i}$$

The left hand side of the entailment must hold, otherwise our objective formula is trivially true.

$$\forall V \exists V^{i_r\prime} \forall V^{\neq i_r\prime} : R^{i_r}_{j_r} \wedge A^{i_r}_{j_r} \wedge \mathbf{0} < \mathrm{RF}^{i_r}_{j_r} \wedge A^{i_r\,\prime}_{j_r} \rightarrow R^{i_r\,\prime}_{j_r} \wedge T^{i_r} \wedge \mathrm{RF}^{i_r\,\prime}_{j_r} < \mathrm{RF}^{i_r}_{j_r} \wedge$$
$$\bigwedge_{i=0,i\neq r}^n \forall V \exists V^{i\prime} \forall V^{\neq i\prime} : R^i_{j_i} \wedge A^i_{j_i} \wedge A^{i\,\prime}_{j_i} \rightarrow R^{i\,\prime}_{j_i} \wedge T^i \wedge \mathrm{RF}^{i\,\prime}_{j_i} = \mathrm{RF}^i_{j_i}$$

If a $\forall V \exists V^{i\prime} \forall V^{\neq i\prime}$ quantified implication holds, then for every assignment to the symbols V such that $R^i_{j_i}(V)$, $A^i_{j_i}(V^{\neq i})$ and, if $i = i_r$, also $\mathbf{0} < \mathrm{RF}^{i_r}_{j_r}(V)$ hold, there exists an assignment to $V^{i\prime}$ satisfying the assumptions of all other E-components $\bigwedge_{s=0,s\neq i}^n A^{s,i}_{j_s}(V^{i\prime})$, for all assignments to the $V^{\neq i\prime}$. Therefore, we can write the following:

$$\forall V \exists \{V^{i\prime}\}_{i=0}^n \forall V^{\neq c\prime} : (R^{i_r}_{j_r} \wedge A^{i_r}_{j_r} \wedge \mathbf{0} < \mathrm{RF}^{i_r}_{j_r} \wedge A^{i_r,\neq c\prime}_{j_r} \rightarrow R^{i_r\,\prime}_{j_r} \wedge T^{i_r} \wedge \mathrm{RF}^{i_r\,\prime}_{j_r} < \mathrm{RF}^{i_r}_{j_r}) \wedge$$
$$\bigwedge_{i=0,i\neq r}^n R^i_{j_i} \wedge A^i_{j_i} \wedge A^{i,\neq c\prime}_{j_i} \rightarrow R^{i\,\prime}_{j_i} \wedge T^i \wedge \mathrm{RF}^{i\,\prime}_{j_i} = \mathrm{RF}^i_{j_i}$$

$\mathbf{0} < \textsc{Rf}^{i_r}_{j_r}(V)$ implies $\mathbf{0} < \textsc{Rf}^{c}_{j}(V)$ and, since $(a \to b) \wedge (c \to d)$ implies $(a \wedge c) \to (b \wedge d)$, the formula above implies:

$$\forall V \exists \{V^{i\prime}\}_{i=0}^{n} \forall V^{\neq c\prime} : (\mathbf{0} < \textsc{Rf}^{c}_{j} \wedge (\bigwedge_{i=0}^{n} R^{i}_{j_i} \wedge A^{i}_{j_i} \wedge A^{i,\neq c\prime}_{j_i})) \to$$
$$\textsc{Rf}^{i_r\,\prime}_{j_r} < \textsc{Rf}^{i_r}_{j_r} \wedge (\bigwedge_{i=0, i\neq i_r}^{n} \textsc{Rf}^{i\,\prime}_{j_i} = \textsc{Rf}^{i}_{j_i}) \wedge \bigwedge_{i=0}^{n} R^{i\,\prime}_{j_i} \wedge T^{i}$$

The formula $\textsc{Rf}^{i_r}_{j_r}(V') < \textsc{Rf}^{i_r}_{j_r}(V)) \wedge (\bigwedge_{i=0, i\neq i_r}^{n} \textsc{Rf}^{i}_{j_i}(V') = \textsc{Rf}^{i}_{j_i}(V))$ implies $\textsc{Rf}^{c}_{j}(V') < \textsc{Rf}^{c}_{j}(V)$ and $\bigwedge_{i=0}^{n} A^{i}_{j_i}(V^{\neq c})$ is equivalent to $A^{c}_{j}(V^{\neq c})$ Therefore, we obtain the implied statement:

$$\forall V \exists \{V^{i\prime}\}_{i=0}^{n} \forall V^{\neq c\prime} :$$
$$((\bigwedge_{i=0}^{n} R^{i}_{j_i} \wedge A^{i}_{j_i}) \wedge A^{c\prime}_{j} \wedge \mathbf{0} < \textsc{Rf}^{c}_{j}) \to ((\bigwedge_{i=0}^{n} T^{i} \wedge R^{i\,\prime}_{j_i}) \wedge \textsc{Rf}^{c\prime}_{j} < \textsc{Rf}^{c}_{j})$$

which is exactly the formula we wanted to prove.

III requires us to prove the following

$$\forall j : 0 \leq j < m^{c} \to$$
$$\exists V, V' : (R^{c}_{j} \wedge A^{c}_{j} \wedge T^{c} \wedge \textsc{Rf}^{c\prime}_{j} = \textsc{Rf}^{c}_{j} \wedge R^{c\prime}_{j} \wedge A^{c\prime}_{j}) \quad \models$$
$$\forall V \exists V^{c\prime} \forall V^{\neq c\prime} : R^{c}_{j} \wedge A^{c}_{j} \wedge A^{c\prime}_{j} \to R^{c\prime}_{j} \wedge T^{c} \wedge \textsc{Rf}^{c\prime}_{j} = \textsc{Rf}^{c}_{j}$$

By definition of $\otimes$ and since $H^{c} \doteq \bigotimes_{i=0}^{n} H^{i}$ we can rewrite it as:

$$\forall \{j_i\}_{i=0}^{n} : \ (\bigwedge_{i=0}^{n} 0 \leq j_i < m^{i}) \to$$
$$\exists V, V' : (\bigwedge_{i=0}^{n} R^{i}_{j_i} \wedge (\bigwedge_{h=0, h\neq i}^{n} A^{i,h}_{j_i}) \wedge A^{i,\neq c}_{j_i} \wedge T^{i}) \wedge \mathit{compatible}_{\{H^{i}\}_{i=0}^{n}} \wedge$$
$$\mathit{indepRank}_{\{H^{i}\}_{i=0}^{n}} \wedge \textsc{Rf}^{c\prime}_{j} = \textsc{Rf}^{c}_{j} \wedge (\bigwedge_{i=0}^{n} R^{i\,\prime}_{j_i} \wedge (\bigwedge_{h=0, h\neq i}^{n} A^{i,h\prime}_{j_i}) \wedge A^{i,\neq c\prime}_{j_i}) \quad \models$$
$$\forall V \exists \{V^{i\prime}\}_{i=0}^{n} \forall V^{\neq c\prime} : ((\bigwedge_{i=0}^{n} R^{i}_{j_i} \wedge (\bigwedge_{h=0, h\neq i}^{n} A^{i,h}_{j_i}) \wedge A^{i,\neq c}_{j_i}) \wedge A^{c\prime}_{j}) \to ((\bigwedge_{i=0}^{n} R^{i\,\prime}_{j_i} \wedge$$
$$(\bigwedge_{h=0, h\neq i}^{n} A^{i,h\prime}_{j_i}) \wedge T^{i}) \wedge \mathit{compatible}_{\{H^{i}\}_{i=0}^{n}} \wedge \mathit{indepRank}_{\{H^{i}\}_{i=0}^{n}} \wedge \textsc{Rf}^{c\prime}_{j} = \textsc{Rf}^{c}_{j})$$

On both sides of the entailment $\textsc{Rf}^c_j(V') = \textsc{Rf}^c_j(V)$ holds, hence $\mathit{indepRank}_{\{H^i\}_{i=0}^n}$ is trivially true: the left-hand-side of the implication in its definition is false. In addition, for any $0 \leq i \leq n$ $A^i_j(V^{\neq c}) \wedge \bigwedge_{h=0,h\neq i}^n A^{i,h}_{j_i}(V^h)$ is equivalent to $A^i_j(V^{\neq i})$. Therefore, our objective formula can be rewritten as:

$$
\begin{aligned}
&\forall \{j_i\}_{i=0}^n : \ (\bigwedge_{i=0}^{n} 0 \leq j_i < m^i) \rightarrow \\
&\exists V, V' : (\bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i} \wedge T^i \wedge {R^i_{j_i}}' \wedge {A^i_{j_i}}') \wedge {\textsc{Rf}^c_j}' = \textsc{Rf}^c_j \wedge \mathit{compatible}_{\{H^i\}_{i=0}^n} \quad \models \\
&\forall V \exists \{V^{i'}\}_{i=0}^n \forall V^{\neq c'} : ((\bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i}) \wedge A^{c\prime}_j) \rightarrow ((\bigwedge_{i=0}^{n} T^i \wedge {R^i_{j_i}}' \wedge \bigwedge_{h=0,h\neq i} A^{i,h'}_{j_i}) \wedge \\
&\quad {\textsc{Rf}^c_j}' = \textsc{Rf}^c_j \wedge \mathit{compatible}_{\{H^i\}_{i=0}^n})
\end{aligned}
$$

If $\mathit{compatible}_{\{H^i\}_{i=0}^n}(V, V')$ does not hold, then the left-hand-side of the entailment is false, hence the entailment is true. Otherwise, $\mathit{compatible}_{\{H^i\}_{i=0}^n}$ holds and since it holds on the left-hand-side of the entailment, it must also hold on the right-hand-side; when both sides of the implication on the right-hand-side of the entailment hold, $\bigwedge_{i=0}^n \bigwedge_{h=0,h\neq i}^n A^{i,h}_{j'_i}(V^{h'})$ must be true since $\mathit{compatible}_{\{H^i\}_{i=0}^n}$ holds. We can further simplify our objective formula as follows:

$$
\begin{aligned}
&\forall \{j_i\}_{i=0}^n : \ (\bigwedge_{i=0}^{n} 0 \leq j_i < m^i) \rightarrow \\
&\quad \exists V, V' : (\bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i} \wedge T^i \wedge {R^i_{j_i}}' \wedge {A^i_{j_i}}') \wedge {\textsc{Rf}^c_j}' = \textsc{Rf}^c_j \wedge \mathit{compatible}_{\{H^i\}_{i=0}^n} \quad \models \\
&\quad \forall V \exists \{V^{i'}\}_{i=0}^n \forall V^{\neq c'} : ((\bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i}) \wedge A^{c\prime}_j)) \rightarrow ((\bigwedge_{i=0}^{n} T^i \wedge {R^i_{j_i}}') \wedge {\textsc{Rf}^c_j}' = \textsc{Rf}^c_j)
\end{aligned}
$$

If the left-hand-side of the entailment is false, then the formula is trivially true. Therefore, assume that there exists a transition performing a self-loop on the restricted region $R^c_j \wedge A^c_j$ in which the ranking function remains constant. Under this assumption, we need to prove the following for any

$j \doteq \langle j_0, \ldots, j_n \rangle$ satisfying the above:

$$\forall V \exists \{V^{i\prime}\}_{i=0}^{n} \forall V^{\neq c\prime} : ((\bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i}) \wedge A^{c\prime}_j) \to (\mathrm{RF}^{c\prime}_j = \mathrm{RF}^c_j \wedge \bigwedge_{i=0}^{n} T^i \wedge R^{i}_{j_i}{}')$$

Hyp. **III** holds for all E-components $\{H^i\}_{i=0}^n$:

$$\begin{aligned}&\forall j_i : 0 \leq j_i < m^i \to \\ &\quad \exists V, V' : (R^i_{j_i} \wedge A^i_{j_i} \wedge T^i \wedge \mathrm{RF}^{i}_{j_i}{}' = \mathrm{RF}^i_{j_i} \wedge R^{i}_{j_i}{}' \wedge A^{i}_{j_i}{}') \quad \models \\ &\quad \forall V \exists V^{i\prime} \forall V^{\neq i\prime} : R^i_{j_i} \wedge A^i_{j_i} \wedge A^{i}_{j_i}{}' \to R^{i}_{j_i}{}' \wedge T^i \wedge \mathrm{RF}^{i}_{j_i}{}' = \mathrm{RF}^i_{j_i}\end{aligned}$$

By assumption there exists a transition in the intersection of their restricted regions such that $\mathrm{RF}^c_j(V') = \mathrm{RF}^c_j(V)$, and hence $\mathrm{RF}^i_{j_i}(V') = \mathrm{RF}^i_{j_i}(V)$ for all i. Therefore, their conjunction implies:

$$\bigwedge_{i=0}^{n} \forall V \exists V^{i\prime} \forall V^{\neq i\prime} : R^i_{j_i} \wedge A^i_{j_i} \wedge A^{i}_{j_i}{}' \to R^{i}_{j_i}{}' \wedge T^i \wedge \mathrm{RF}^{i}_{j_i}{}' = \mathrm{RF}^i_{j_i}$$

If a $\forall V \exists V^{i\prime} \forall V^{\neq i\prime}$ quantified implication holds then for every assignment to the symbols V such that $R^i_{j_i}(V) \wedge A^i_{j_i}(V^{\neq i}) \wedge A^i_{j_i}(V^{\neq i\prime})$ holds, there exists an assignment to the $V^{i\prime}$ satisfying the assumptions of all other E-components $\bigwedge_{s=0, s\neq i}^{n} A^{s,i}_{j_s}(V^{i\prime})$, for all assignments to the $V^{\neq i\prime}$. Therefore, we can write the following:

$$\forall V \exists \{V^{i\prime}\}_{i=0}^{n} \forall V^{\neq c\prime} : \bigwedge_{i=0}^{n} ((R^i_{j_i} \wedge A^i_{j_i} \wedge A^{i,\neq c\prime}_{j_i}) \to (R^{i}_{j_i}{}' \wedge T^i \wedge \mathrm{RF}^{i}_{j_i}{}' = \mathrm{RF}^i_{j_i}))$$

Since $(a \to b) \wedge (c \to d)$ implies $(a \wedge c) \to (b \wedge d)$ and $\bigwedge_{i=0}^{n} \mathrm{RF}^i_{j_i}(V') = \mathrm{RF}^i_{j_i}(V)$ implies $\mathrm{RF}^c_j(V') = \mathrm{RF}^c_j(V)$, the formula above implies:

$$\forall V \exists \{V^{i\prime}\}_{i=0}^{n} \forall V^{\neq c\prime} : (\bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i}) \wedge A^{c\prime}_j \to (\mathrm{RF}^{c\prime}_j = \mathrm{RF}^c_j \wedge \bigwedge_{i=0}^{n} T^i \wedge R^{i}_{j_i}{}')$$

which is exactly the formula we wanted to prove.

IV requires us to prove the following

$$\begin{aligned}&\forall j, j' :\ 0 \leq j < m^c \wedge 0 \leq j' < m^c \to \\ &\exists V, V' : (R^c_j \wedge A^c_j \wedge T^c \wedge \mathrm{RF}^c_j = \mathbf{0} \wedge R^{c}_{j'}{}' \wedge A^{c}_{j'}{}') \quad \models \\ &\forall V \exists V^{c\prime} \forall V^{\neq c\prime} : R^c_j \wedge A^c_j \wedge \mathrm{RF}^c_j = \mathbf{0} \wedge A^{c}_{j'}{}' \to R^{c}_{j'}{}' \wedge T^c\end{aligned}$$

By definition of $\otimes$ and since $H^c \doteq \bigotimes_{i=0}^n H^i$ we can rewrite it as:

$$\forall \{j_i\}_{i=0}^n, \{j_i'\}_{i=0}^n : (\bigwedge_{i=0}^n 0 \leq j_i < m^i \wedge 0 \leq j_i' < m^i) \rightarrow$$

$$\exists V, V' : (\bigwedge_{i=0}^n R_{j_i}^i \wedge (\bigwedge_{h=0,h\neq i}^n A_{j_i}^{i,h}) \wedge A_{j_i}^{i,\neq c} \wedge T^i) \wedge \mathit{compatible}_{\{H^i\}_{i=0}^n} \wedge$$

$$\mathit{indepRank}_{\{H^i\}_{i=0}^n} \wedge \mathrm{RF}_j^c = \mathbf{0} \wedge (\bigwedge_{i=0}^n {R_{j_i'}^i}' \wedge (\bigwedge_{h=0,h\neq i}^n {A_{j_i'}^{i,h}}') \wedge {A_{j_i'}^{i,\neq c}}') \quad \models$$

$$\forall V \exists \{{V^i}'\}_{i=0}^n \forall {V^{\neq c}}' : ((\bigwedge_{i=0}^n R_{j_i}^i \wedge (\bigwedge_{h=0,h\neq i}^n A_{j_i}^{i,h}) \wedge A_{j_i}^{i,\neq c}) \wedge \mathrm{RF}_j^c = \mathbf{0} \wedge {A_{j'}^c}') \rightarrow$$

$$((\bigwedge_{i=0}^n {R_{j_i'}^i}' \wedge T^i \wedge \bigwedge_{h=0,h\neq i}^n {A_{j_i'}^{i,h}}') \wedge \mathit{compatible}_{\{H^i\}_{i=0}^n} \wedge \mathit{indepRank}_{\{H^i\}_{i=0}^n})$$

If $j \neq j'$, $\mathit{indepRank}_{\{H^i\}_{i=0}^n}$ trivially holds, since the left-hand-side of the implication in its definition is false. Otherwise, if $j = j'$, $\mathrm{RF}_j^c(V) = \mathbf{0}$ contradicts $\mathrm{RF}_j^c(V') < \mathrm{RF}_j^c(V)$ and again $\mathit{indepRank}_{\{H^i\}_{i=0}^n}$ trivially holds because the left-hand-side of the implication in its definition is false. In addition, for any $0 \leq i \leq n$ $A_j^i(V^{\neq c}) \wedge \bigwedge_{h=0,h\neq i}^n A_{j_i}^{i,h}(V^h)$ is equivalent to $A_j^i(V^{\neq i})$. Therefore, our objective formula can be rewritten as:

$$\forall \{j_i\}_{i=0}^n, \{j_i'\}_{i=0}^n : (\bigwedge_{i=0}^n 0 \leq j_i < m^i \wedge 0 \leq j_i' < m^i) \rightarrow$$

$$\exists V, V' : (\bigwedge_{i=0}^n R_{j_i}^i \wedge A_{j_i}^i \wedge T^i \wedge {R_{j_i'}^i}' \wedge {A_{j_i'}^i}') \wedge \mathit{compatible}_{\{H^i\}_{i=0}^n} \wedge \mathrm{RF}_j^c = \mathbf{0} \quad \models$$

$$\forall V \exists \{{V^i}'\}_{i=0}^n \forall {V^{\neq c}}' : ((\bigwedge_{i=0}^n R_{j_i}^i \wedge A_{j_i}^i) \wedge \mathrm{RF}_j^c = \mathbf{0} \wedge {A_{j'}^c}') \rightarrow$$

$$((\bigwedge_{i=0}^n T^i \wedge {R_{j_i'}^i}' \wedge \bigwedge_{h=0,h\neq i}^n {A_{j_i'}^{i,h}}') \wedge \mathit{compatible}_{\{H^i\}_{i=0}^n})$$

If $\mathit{compatible}_{\{H^i\}_{i=0}^n}(V, V')$ does not hold, then the left-hand-side of the entailment is false, hence the entailment is true. Otherwise $\mathit{compatible}_{\{H^i\}_{i=0}^n}$ holds and since it holds on the left-hand-side of the entailment, it must also hold on the right-hand-side; when both sides of the implication on the right-hand-side of the entailment hold, $\bigwedge_{i=0}^n \bigwedge_{h=0,h\neq i}^n A_{j_i'}^{i,h}({V^h}')$ must be true since $\mathit{compatible}_{\{H^i\}_{i=0}^n}$ holds. We can further simplify our objective formula as follows:

$$\forall \{j_i\}_{i=0}^{n}, \{j'_i\}_{i=0}^{n} : (\bigwedge_{i=0}^{n} 0 \le j_i < m^i \wedge 0 \le j'_i < m^i) \rightarrow$$

$$\exists V, V' : (\bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i} \wedge T^i \wedge {R^i_{j'_i}}' \wedge {A^i_{j'_i}}') \wedge compatible_{\{H^i\}_{i=0}^{n}} \wedge \mathrm{RF}^c_j = \mathbf{0} \quad \models$$

$$\forall V \exists \{V^{i'}\}_{i=0}^{n} \forall V^{\neq c'} : (\mathrm{RF}^c_j = \mathbf{0} \wedge \bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i} \wedge A^{i,\neq c'}_{j'_i}) \rightarrow (\bigwedge_{i=0}^{n} T^i \wedge {R^i_{j'_i}}')$$

If the left-hand-side of the entailment is false, then the formula is trivially true. Therefore, assume that there exists a transition from a state in $R^c_j \wedge A^c_j \wedge \mathrm{RF}^c_j = \mathbf{0}$ to $R^c_{j'} \wedge A^c_{j'}$ Under this assumption, we need to prove the following for any $j \doteq \langle j_0, \ldots, j_n \rangle$ satisfying the above:

$$\forall V \exists \{V^{i'}\}_{i=0}^{n} \forall V^{\neq c'} : (\mathrm{RF}^c_j = \mathbf{0} \wedge \bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i} \wedge A^{i,\neq c'}_{j'_i}) \rightarrow (\bigwedge_{i=0}^{n} T^i \wedge {R^i_{j'_i}}')$$

Each E-component H^i allows for a transition from its restricted region with index j_i to the one with index j'_i. In this transition since $\mathrm{RF}^c_j(V) = \mathbf{0}$, then $\mathrm{RF}^i_{j_i}(V) = \mathbf{0}$ holds in the source state. The following holds since Hyp. **IV** holds for all $\{H^i\}_{i=0}^{n}$.

$$\exists V, V' : (R^i_{j_i} \wedge A^i_{j_i} \wedge T^i \wedge \mathrm{RF}^i_{j_i} = \mathbf{0} \wedge {R^i_{j'_i}}' \wedge {A^i_{j'_i}}') \quad \models$$
$$\forall V \exists V^{i'} \forall V^{\neq i'} : (R^i_{j_i} \wedge A^i_{j_i} \wedge \mathrm{RF}^i_{j_i} = \mathbf{0} \wedge {A^i_{j'_i}}') \rightarrow {R^i_{j'_i}}' \wedge T^i$$

If a $\forall V \exists V^{i'} \forall V^{\neq i'}$ quantified implication holds then for every assignment to the symbols V such that $R^i_{j_i}(V) \wedge A^i_{j_i}(V^{\neq i}) \wedge A^i_{j'_i}(V^{\neq i'})$ holds, there exists an assignment to the $V^{i'}$ satisfying the assumptions of all other E-components $\bigwedge_{s=0, s\neq i}^{n} A^{s,i}_{j'_s}(V^{i'})$, for all assignments to the $V^{\neq i'}$. Therefore, we can write the following:

$$\forall V \exists \{V^{i'}\}_{i=0}^{n} \forall V^{\neq c'} : \bigwedge_{i=0}^{n} ((R^i_{j_i} \wedge A^i_{j_i} \wedge \mathrm{RF}^i_{j_i} = \mathbf{0} \wedge A^{i,\neq c'}_{j'_i}) \rightarrow ({R^i_{j'_i}}' \wedge T^i))$$

Since $(a \rightarrow b) \wedge (c \rightarrow d)$ implies $(a \wedge c) \rightarrow (b \wedge d)$ and $\bigwedge_{i=0}^{n} \mathrm{RF}^i_{j_i}(V) = \mathbf{0}$ implies $\mathrm{RF}^c_j(V) = \mathbf{0}$, we can write the following implied statement:

$$\forall V \exists \{V^{i'}\}_{i=0}^{n} \forall V^{\neq c'} : (\mathrm{RF}^c_j = \mathbf{0} \wedge \bigwedge_{i=0}^{n} R^i_{j_i} \wedge A^i_{j_i} \wedge A^{i,\neq c'}_{j'_i}) \rightarrow (\bigwedge_{i=0}^{n} T^i \wedge {R^i_{j'_i}}')$$

which is exactly the formula we wanted to prove.

□

Appendix B.5. E-CHC encoding of funnel-loop search is sound

Theorem 7. *Given a fair transition system $M \doteq \langle V, I^M, T^M, F^M \rangle$ and an interpretation for the queries R, T and Rank satisfying all Eqs. (A.1)–(A.7). Then there exist a funnel-loop for M.*

Proof. We first show that, $R(c, V)$ and $T(V, V')$ correspond to a funnel and then show that such funnel corresponds to a funnel-loop of length one. We define a funnel $fnl \doteq \langle S(V), T_{fnl}(V, V'), D(V), \textsc{Rf}(V) \rangle$, where (i) $S(V) \doteq \exists c : R(c, V)$, (iv) $T_{fnl}(V, V') \doteq T(V, V') \wedge (D(V') \leftrightarrow \textsc{Rf}(V) = \mathbf{0})$, (iii) $D(V) \doteq \exists c : c \wedge R(c, V)$ and (iii) $\textsc{Rf}(V)$ is a ranking function witnessing the well-foundedness of relation $Rank(V, V')$. $\textsc{Rf}$ is such that $\textsc{Rf}(V) = 0$ for all V such that there exist no V' making $\neg c \wedge R(c, V) \wedge T(V, V') \wedge \wedge c' R(c', V')$ hold:

$$\forall c, V, c' : \neg(\exists V' : \neg c \wedge R(c, V) \wedge T(V, V') \wedge c' \wedge R(c', V')) \rightarrow \textsc{Rf}_i(V) = 0$$

and in all other cases $(R(c, V) \wedge T(V, V') \wedge R(c, V')$ holds for all V, $V')$ the following must hold:

$$\forall V, V' : (\neg c \wedge R(c, V) \wedge T(V, V') \wedge R(c, V')) \rightarrow \textsc{Rf}(V) \geq \textsc{Rf}(V') + 1$$

These two constraints allow for many different interpretations of $\textsc{Rf}$. Every such interpretation satisfies our requirements and it is sufficient for such set to be non-empty. The well-foundedness of *Rank* implies, by Eq. (A.6), that $\neg c \wedge R(c, V) \wedge T(V, V') \wedge R(c, V')$ is well-founded. Therefore, there must exist some V such that $\textsc{Rf}(V) = 0$: in particular all the states in $\neg c \wedge \neg R(c, V)$ and all the states in $\neg c \wedge R(c, V)$ for which T does not admit any successor in the same region. Since $\neg c \wedge R(c, V) \wedge T(V, V') \wedge R(c, V')$ is well-founded it cannot allow for any infinite chain of states, hence it cannot allow any loop of states. Therefore, the constraints above do not contain any circular dependency in the definition of the assignments to the $\textsc{Rf}(V)$ and there exists at least one interpretation for $\textsc{Rf}$.

We now show that fnl satisfies all hypotheses required by Def. 1.

F.1 follows directly from Eq. (A.4) and the fact that $\textsc{Rf} = \mathbf{0}$ implies that T does not admit any successor in $\neg c \wedge R(c, V)$, hence it must admit some successor in $c \wedge R(c, V)$, which by definition is in D.

F.2 By construction S contains all states of $\exists c : R(c, V)$. Eq. (A.2) ensures that this is an invariant, hence Hyp. F.2 holds.

F.3 By construction, $\textsc{Rf}$ assigns decreasing integers to the chains described by the relation $\neg c \wedge R(c, V) \wedge T(V, V') \wedge R(c, V')$. Therefore, at every such step $\textsc{Rf}$ must decrease and Hyp. F.3 holds.

F.4 Eq. (A.2) and the well-foundedness of $\neg c \wedge R(c, V) \wedge T(V, V')$, ensures that from a state in $\neg c \wedge R(c, V)$ in a finite number of T steps we must reach a state in $c \wedge R(c, V)$. We defined $\textsc{Rf}$ such that $\textsc{Rf} = \mathbf{0}$ in the states whose T successors are in $c \wedge R(c, V)$, hence in D. Therefore, Hyp. F.4 holds.

We now show that fnl is a funnel-loop: it meets all hypotheses of Def. 2

FL.1 trivially holds since fnl is the only funnel.

FL.2 We defined S as the union of $c \wedge R(c, V)$ and $\neg c \wedge R(c, v)$ and D as $c \wedge R(c, V)$. Therefore $D \rightarrow S$ and Hyp. FL.2 holds.

Finally, we show that this funnel-loop represents at least one fair path of M by showing that it meets all hypotheses of Th. 1.

FF.1 holds since Eq. (A.1) ensures that $R(c, V)$ has a non-empty intersection with the initial states I^M.

FF.2 holds since Eq. A.5 ensures that every state in $c \wedge R(c, V)$ satisfies $F^M(V)$. We defined $D \doteq R(\top, V)$, hence $D \rightarrow F^M$ and Hyp. FF.2 must hold.

FF.3 follows directly from Eq. (A.3).

□

Appendix B.6. E-CHC encoding of funnel-loop search is complete

Theorem 8. *Let $floop$ be a funnel-loop of length one for a transition system $M \doteq \langle V, I^M, T^M, F^M \rangle$. Then, there exists an intepretation for the query symbols R, T and $Rank$ satisfying all Eqs.* (A.1)–(A.7).

Proof. Given a $floop$ of length one, we define an interpretation for the query symbols R, T and $Rank$ for the E-CHC. Let $fnl \doteq \langle S, T_{fnl}, D, \text{R}\textsc{f} \rangle$ be the funnel of $floop$. By Th. 1 there exists a finite sequence of states π such that: it starts from an initial state of M, follows the transition relation of M and ends in a state in the source region S. Without loss of generality we assume π does not contain any state in S other than the last one. In the following we write $\pi(V)$ for the predicate that holds iff V is in π and $\pi(V, V')$ for the predicate that holds iff V and V' are two consecutive states in π. We define the interpretation for the queries as follows: (i) $R(c, V) \doteq (\pi(V) \vee S(V)) \wedge (c \leftrightarrow D(V))$, (ii) $T(V, V') \doteq \pi(V, V') \vee (S(V) \wedge T_{fnl}(V, V'))$ and (iii) $Rank(V, V') \doteq \pi(V, V') \vee \text{R}\textsc{f}(V') < \text{R}\textsc{f}(V)$. We now show that this interpretation satisfies all Eqs. (A.1)–(A.7).

Eq. (A.1) By construction $\neg c \wedge R(c, V)$ contains all states in π. By hypothesis, the first state of π is an initial state of M. Therefore, Eq. (A.1) holds.

Eq. (A.2) $R(c, V)$ contains all states of π and of S. T either follows the transitions of π or, once it reaches S follows the transition relation of fnl. By hypotheses F.2, F.4, FL.1 and FL.2 such transitions must remain in S. Therefore, from every state not in S and not in π T is false and the left-hand-side of Eq. (A.2) is false; otherwise, every T transition must remain within $R(c, V)$ and Eq. (A.2) is true.

Eq. (A.3) Every step in π is also a step in M and by Hyp. FF.3 every step of $floop$ underapproximates the transition relation of M. Therefore, $T(V, V')$ underapproximates T^M and Eq. (A.3) holds.

Eq. (A.4) Since Hyp. F.1 must hold for fnl and every state in π must admit a successor until a state in S is reached, by construction $T(V, V')$ always allows from some successor state in each region $R(V, c)$. Therefore, Eq. (A.4) holds.

Eq. (A.5) By Hyp. FF.2 the destination region D underapproximates the fair states. By construction $c \wedge R(c, V)$ is equivalent to such region. Therefore, Eq. A.5 holds.

Eq. (A.6) holds by construction of the interpretation for *Rank*.

Eq. (A.7) holds since π is a finite sequence of states and each RF is a ranking function with respect to T_{fnl} and the corresponding S.

$\square$