(Asynchronous) Temporal Logics for Hyperproperties on Finite Traces

Alberto Bombardelli[1,2], Laura Bozzelli[3], Cesar Sanchez[4], and Stefano Tonetta[1]

[1] Fondazione Bruno Kessler, Via Sommarive, 18, 38123 Povo TN, Italy
[2] University of Trento, Via Sommarive, 9, 38123 Povo TN, Italy
[3] University of Napoli "Federico II", Napoli, Italy
[4] IMDEA Software Institute, Pozuelo de Alarcón, Madrid, Spain

Abstract. Hyperproperties are properties of systems that relate more than one trace, which are common in information-flow, concurrency, symmetry, diagnosability, etc. Several temporal logics for hyperproperties have been proposed—like HyperLTL and HyperCTL* where traces are compared synchronously—and recently extended for comparing traces asynchronously. However, all these temporal hyperlogics are for infinite traces, in spite of the recent popularity of temporal logics on finite traces. In this paper, we introduce SC-HyperLTL, a logic for asynchronous hyperproperties on *finite traces* and *with past operators*, by adapting and combining two existing logics for asynchronous hyperproperties: stuttering HyperLTL (HyperLTL$_S$) and context HyperLTL (HyperLTL$_C$). We show that SC-HyperLTL is decidable and can encode many hyperproperties of interest. We present two practical algorithms that reduce the model checking problem of SC-HyperLTL to inputs of two existing tools: AutoHyper (which can model check hyperproperties on infinite traces) and nuXmv (a model checker for trace properties). We illustrate our algorithms on a prototype implementation.

1 Introduction

Hyperproperties [18] are properties of systems that relate more than one trace at a time, in contrast with trace properties whose semantics are defined on a single trace. Hyperproperties are common in information-flow security [26,38], robustness and sensitivity of cyber-physical systems [37], concurrency [12] and symmetry [23]. HyperLTL [17] is a temporal logic for hyperproperties that equips LTL with quantifiers for different traces. HyperLTL includes relational capabilities to compare different traces and temporal modalities where the temporal operators move all traces synchronously. Several algorithms and verification methods for HyperLTL have been proposed [23,20,30].

In settings like software verification execution traces are finite and of varying length, and relevant events happen at different instants. Therefore, the semantics of the temporal operators need to cope with when traversing the different traces at different speeds. Several recent papers [28,3,14,15,1] have introduced hyperlogics to capture asynchrony between the traces. However, all these temporal logics for hyperproperties, synchronous and asynchronous, consider only *infinite traces* and do not have *past operators*.

At the same time, there is a growing interest in linear temporal logics for finite traces [32,22,25] because of their applications to AI, synthesis and runtime verification. An obstacle for a temporal hyper-logic on finite traces is that, even though the beginning of the trace is a point of synchrony between all traces, the end of the trace may arrive at different instants. This is similar when traversing the trace backwards with past operators.

In this paper we introduce SC-HyperLTL, a novel temporal logic for hyperproperties on finite traces that combines modalities from two known asynchronous logics [14,15] for infinite traces: stuttering HyperLTL (HyperLTL$_S$) and context HyperLTL (HyperLTL$_C$). HyperLTL$_S$ allows to stutter on each trace individually and define points of evaluation that can be arbitrarily apart, while HyperLTL$_C$ allows to restrict which traces proceed according to the temporal operators. We prove that SC-HyperLTL can express many properties of interest and its model-checking is decidable. Then, we present two approaches for the practical model checking $M \models \varphi$ of our logic.

1. We first show a stuttering translation that modifies M to account for individual trace stuttering, and introduces additional constraints to φ to align the traces. The result is a HyperLTL model-checking problem whose output correspond to $M \models \varphi$. This approach only works for quantifier alternation free formulas.
2. The second algorithm introduces k history variables and equips the formula with constraints—local to each trace—for the correctness of the auxiliary variables. This approach works for arbitrary quantifier alternation, and can verify systems where sequences of k observations, which can be arbitrarily far apart, are sufficient to prove or disprove the hyperproperty.

In summary, the contributions of this paper are: (1) A novel logic SC-HyperLTL for (asynchronous) hyperproperties on finite traces with past operators, introduced in Section 3. (2) Two practical model checking algorithms for SC-HyperLTL, presented in Section 4. (3) A report on a proof-of-concept implementation of these algorithms, reported in Section 5.

Motivating Example. Consider the following terminating function P, with input *in* and output *out* which iteratively set *out* to $10(in + 1)$.

```
int tmp = 1;
out = in;
for (int i=0;i<10;i++){
  int j = 0;
  while (j++ < in) tmp++;
  out = tmp;
}
```

We would like to show that—in spite of the internal program dynamics—if *in* is smaller at the beginning of the execution, the n-th output will be also smaller during the whole execution. We call this property *reactive monotonicity*, which requires to relate multiple executions of the same program and, since the program has to terminate, to reason with finite semantics. One attempt using a variation of HyperLTL with finite semantics is:

$$\varphi_{rmon}^{sync} := \forall x.\forall y.in[x] < in[y] \rightarrow \mathbf{G}(out[x] < out[y])$$

which expresses that, for each pair of executions of P (here represented by trace variables x and y) if x starts with an input lower than that of y ($in[x] < in[y]$), then it is always the case that the output of x is smaller than the output of y: $\mathbf{G}(out[x] < out[y])$. During its execution, the program might perform internal computations that are not relevant to the desired property but that can change the outcome due to the synchronous alignment. Therefore, even if P guarantees that the same assignment sequences of the variable *out* are "reactive monotonic" in two executions, the property can be violated:

$$\sigma_1 := \cdots \rightarrow (j = 2, l = 5, out = 2, in = 2) \rightarrow (j = 3, l = 6, out = 1) \rightarrow (out = 3)$$
$$\sigma_2 := \cdots \rightarrow (j = 2, l = 5, out = 3, in = 3) \rightarrow (j = 3, l = 5, out = 1) \rightarrow (out = 3)$$

To fix this excessive synchrony, we adopt a semantics that evaluates the traces *asynchronously* in general but *synchronously* over a set of interesting points. In our example, we want to check the values of *out* at those points in which *out* changes, which is achieved with the *stuttering* variants of the temporal modalities. We also introduce *context* temporal modalities to reason over a subset of the traces in a formula, restricting the temporal progress to those traces. The use of contexts in this example is also motivated by the additional challenge that finite traces introduce. The resulting formula is:

$$\varphi_{rmon}^{SC} := \forall x.\forall y.\{out\}.in[x] < in[y] \rightarrow \mathbf{G}((end[x] \leftrightarrow end[y]) \wedge out[x] < out[y])$$

where $end[x]$ captures whether trace x has a successor, defined as $\langle x \rangle \mathbf{X} \top$ using the context modality. The set $\{out\}$ denotes that the interesting positions are the points in which *out* changes. The globally modality is then evaluated only over interesting positions i.e. where *out* changes. Finally, φ_{rmon}^{SC} uses singleton contexts with next operator in $end[x]$ and $end[y]$ to ensure that the traces have the same amount of "interesting points", by requiring that x has a successor iff y has a successor.

Related Work. Starting from HyperLTL and HyperCTL* [17], many other decidable temporal logics for hyperproperties have been proposed, including HyperQPTL [35,19] and HyperPDL$-\Delta$ [27]. The semantics of all these logics are synchronous, meaning that temporal modalities move all traces in a lock-step manner. Moreover, none of these logics consider past operators.

Recently, several works have introduced hyperlogics capable of handling an asynchronous traversal of traces, which is useful for addressing speed variations introduced by compiler optimizations, scheduling, and other factors. In [28], the temporal fixpoint calculus H_μ is introduced, featuring operators to describe the independent progress of each trace. In [5], the asynchronous hyperlogic Observation HyperLTL is introduced. This logic is decidable and similar to the one defined here. The main differences between the two are that (*i*) SC-HyperLTL allows reasoning locally on traces using singleton contexts, and (*ii*) observational HyperLTL considers different observation conditions for each trace, whereas our logic selects a set of LTL formulas to define identical observations for all traces. Similarly, [3] presents *Asynchronous HyperLTL* (A-HLTL), which extends HyperLTL with the notion of trajectories, introduced in [11], to control the progress of the traces. Our first algorithm is similar to the stutter algorithm from [3], but our fragment is strictly richer, allowing synchronous bounded operators as well and directly supporting past operators. Two other extensions of HyperLTL [14] are stuttering HyperLTL (HyperLTL$_S$), which describes with LTL formulas the interesting points of each trace, and context HyperLTL (HyperLTL$_C$), which restricts the progress of specific subsets of traces. In a recent work [8], we defined a general combination of HyperLTL$_S$ and HyperLTL$_C$ for infinite traces with non-prenex quantifiers. The logic presented in that work subsumes the logic considered in this paper. However, that work was purely theoretical, focusing on a general logic and a decidable fragment to handle complex diagnosability specifications such as Finite Delay Diagnosability [13]. All of these logics are defined for infinite traces and most of them do not support past operators. Two exception for finite traces are [24] and [2]. For what regards [24], the logic introduced there is synchronous, lacks past operators, and provides no tool. On the other hand, in [2] the formalism considers both finite traces and asynchronous systems. However, the formalism is conceptually different and based on automata rather than temporal logics.

Several tools have been proposed for verifying synchronous hyperproperties: MCHyper [23] (which reduces alternating-free verification to LTL model checking), AutoHyper [6] (which implements explicit-state automata constructions), and HyperQB [30] (which follows a bounded-model approach by generating queries for a QBF solver). Recently, new works have focused on verifying asynchronous hyperlogics [29,5,3,4,21]. In [29], the BMC-based verification of HyperLTL has been adapted to handle A-HLTL. While similar in spirit to our second algorithm, the key difference lies in our algorithm's ability to reason over unbounded executions. Another notable difference between the two approaches is in their reduction techniques. Our method reduces the problem to a synchronous HyperLTL model-checking problem, whereas [29] reduces it to a QBF verification problem. The remaining works employ very different techniques. In [5], the authors reduce Observation HyperLTL model checking to a game-based verification using predicate abstraction. In [21], symbolic execution is used to efficiently find counterexamples to asynchronous $\forall\exists$ hyperproperties. Finally, in [4], Hoare logic is used to reason over $\forall\exists$ temporal safety properties in asynchronous programs.

2 Preliminaries

We use $\mathbb{N}$ for the set of natural numbers, (i, j) for the set $\{k \mid i < k < j\}$, $[i, j]$ for $\{k \mid i \leq k \leq j\}$, $[i, j)$ for $\{k \mid i \leq k < j\}$ and $(i, j]$ for $\{k \mid i < k \leq j\}$. We fix a finite set AP of atomic propositions and use $\Sigma = 2^{\mathsf{AP}}$ for the alphabet. A finite trace is a finite sequence $\sigma = a_0 a_1 \cdots$ of letters from Σ is (i.e. a word from Σ^*). Similarly, an infinite trace σ is a word from Σ^ω. Given a finite or infinite word σ, we use $\sigma(i)$ for a_i and σ^i for the suffix $a_i a_{i+1} \cdots$. A *pointed trace* is a pair (σ, i), where $i \in \mathbb{N}$ is a natural number (called the *pointer*). Given a pointed trace (σ, i) and $n \in \mathbb{N}$, we use $(\sigma, i) + n$ for $(\sigma, i + n)$.

Symbolic Transition System. A *Symbolic Transition System* (STS) M is a tuple $M = \langle V, I, T, F\rangle$ where V is a set of Boolean (state) variables, $I(V)$ is a formula representing the initial states, $T(V, V')$ is a formula representing the transitions, and F is a set of formulae $f(V)$ representing the fairness conditions (final condition on finite traces) i.e. a condition that must occur infinitely often in an infinite trace (at the end of finite traces). A *state* of M is an assignment to the variables V. A *trace* of M is an infinite sequence $s_0, s_1, \ldots$ of states such that $s_0 \models I$, for all $i \geq 0$, $s_i, s'_{i+1} \models T$, and for all $f \in F$, for all $i \geq 0$ there exists $j \geq i$ s.t. $s_j \models f$. Similarly, a finite trace $s_0, s_1, \ldots$ is a finite sequence such that $s_0 \models I$, for all $i \geq 0$, $s_i, s'_{i+1} \models T$, and for all $f \in F$: $(\sigma, |\sigma| - 1) \models f$. Given two transitions systems $M_1 = \langle V_1, I_1, T_1, F_1\rangle$ and $M_2 = \langle V_2, I_2, T_2, F_2\rangle$, we denote with $M_1 \times M_2$ the synchronous product $\langle V_1 \cup V_2, I_1 \wedge I_2, T_1 \wedge T_2, F_1 \cup F_2\rangle$. We refer to $\mathsf{Traces}(M)$ as the set of all infinite traces of M and $\mathsf{Traces}^{fin}(M)$ for the set of non-empty finite traces of M.

The Temporal Logics LTL and LTL$_f$. We consider LTL [34] with past [31] (called PLTL), which has the following syntax:

$$\varphi ::= \top \mid a \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\varphi \mid \mathbf{Y}\varphi \mid \varphi\, \mathbf{U}\, \varphi \mid \varphi\, \mathbf{S}\, \varphi$$

where a is a proposition, $\vee$ and $\neg$ are the usual Boolean disjunction and negation, while $\mathbf{X}$, $\mathbf{U}$, $\mathbf{Y}$ and $\mathbf{S}$ are respectively the "next", "until", "yesterday" and "since" temporal operators. The semantics of PLTL associates pointed traces with formulae as follows (we omit Boolean connectives, which are standard):

$$
\begin{array}{ll}
(\sigma, i) \models a & \Leftrightarrow a \in \sigma(i) \\
(\sigma, i) \models \mathbf{X}\varphi & \Leftrightarrow i < |\sigma| - 1 \text{ and } (\sigma, i+1) \models \varphi \\
(\sigma, i) \models \mathbf{Y}\varphi & \Leftrightarrow i > 0 \text{ and } (\sigma, i-1) \models \varphi \\
(\sigma, i) \models \varphi_1\, \mathbf{U}\, \varphi_2 & \Leftrightarrow \text{for some } j \in [i, |\sigma|), (\sigma, j) \models \varphi_2 \text{ and for all } k \in [i, j), (\sigma, k) \models \varphi_1 \\
(\sigma, i) \models \varphi_1\, \mathbf{S}\, \varphi_2 & \Leftrightarrow \text{for some } j \in [0, i], (\sigma, j) \models \varphi_2 \text{ and for all } k \in (j, i], (\sigma, k) \models \varphi_1
\end{array}
$$

We also use common derived operators like $\bot$, $\wedge$, $\mathbf{R}$ (dual w.r.t. negation of $\mathbf{U}$), $\mathbf{F}$, $\mathbf{G}$, $\mathbf{O}$ ($\mathbf{O}\phi := \top\, \mathbf{S}\, \phi$), $\mathbf{H}$ (dual w.r.t. negation of $\mathbf{O}$), $\mathbf{T}$ (dual w.r.t. negation of $\mathbf{S}$) and $\mathbf{Z}$ (the dual w.r.t. negation of $\mathbf{Y}$). The variant LTL$_f$ of PLTL for finite traces [33,25] borrows the syntax from LTL, with an additional abbreviation:

$\mathbf{N}\varphi = \neg\mathbf{X}\neg\varphi$. A Symbolic Transition System M is a model of an PLTL formula φ, denoted by $M \models \varphi$, whenever $\mathsf{Traces}(M) \models \varphi$. Similarly, M models an LTL$_f$ formula φ whenever $\mathsf{Traces}^{fin}(M) \models \varphi$.

The Temporal Logic HyperLTL. HyperLTL [17] is a temporal logic for hyperproperties. The syntax of HyperLTL is:

$$\alpha ::= \exists x.\alpha \mid \forall x.\alpha \mid \varphi \qquad\qquad \varphi ::= a[x] \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\varphi \mid \varphi\,\mathbf{U}\,\varphi$$

where x is a *trace variable* from an infinite set VAR of trace variables. The intended meaning of $a[x]$ is that proposition a holds at the current time in trace x. Trace quantifiers $\exists x$ and $\forall x$ allow reasoning simultaneously about different traces of the computation. Atomic predicates $a[x]$ refer to a single trace x. Given an PLTL formula α, we denote by $\alpha[x]$ the formula obtained by substituting every atomic proposition a with $a[x]$.

Given a HyperLTL formula φ, we use $\mathsf{TrVars}(\varphi)$ for the set of trace variables quantified in φ. A formula φ is well-formed if for all atoms $a[x]$, x is quantified in φ. Given a set of infinite traces T, the semantics of a HyperLTL formula α is defined in terms of *pointed trace assignments*, which are partial mappings of the form $\Pi : \mathsf{TrVars}(\alpha) \rightharpoonup (T \times \mathbb{N})$. We use $Dom(\Pi)$ for the subset of $\mathsf{TrVars}(\varphi)$ for which Π is defined. Given a trace assignment Π, a trace variable x, a trace σ and a pointer p, we denote by $\Pi[x \mapsto (\sigma, p)]$ the assignment that coincides with Π for every trace variable except for x, which is mapped to (σ, p). The trace assignment with empty domain is denoted by Π_0. Also, we use $\Pi + n$ to denote trace assignment Π' such that $\Pi'(x) = \Pi(x) + n$ for all $x \in Dom(\Pi) = Dom(\Pi')$. Given a partial trace assignment Π and a set of traces T, the semantics of HyperLTL is as follows (we again omit $\vee$ and $\neg$):

$$\begin{array}{ll}
\Pi \models_T \exists x.\alpha & \Leftrightarrow \text{for some } \sigma \in T,\ \Pi[x \mapsto (\sigma,0)] \models_T \alpha \\
\Pi \models_T \forall x.\alpha & \Leftrightarrow \text{for all } \sigma \in T,\ \Pi[x \mapsto (\sigma,0)] \models_T \alpha \\
\Pi \models_T \varphi & \Leftrightarrow \Pi \models \varphi \\
\Pi \models a[x] & \Leftrightarrow a \in \sigma(p),\ \text{ where } (\sigma,p) = \Pi(x) \\
\Pi \models \varphi_1\,\mathbf{U}\,\varphi_2 & \Leftrightarrow \text{for some } j \geq 0\ \ (\Pi + j) \models \varphi_2, \text{and forall } i < j, (\Pi + i) \models \varphi_1 \\
\Pi \models \mathbf{X}\varphi & \Leftrightarrow (\Pi + 1) \models \varphi
\end{array}$$

Note that quantifiers assign traces to trace variables and set the pointer to the initial position 0. We say that a set of infinite traces T is a model of a HyperLTL formula φ, denoted $T \models \varphi$ whenever $\Pi_0 \models_T \varphi$. We say that a Symbolic Transition System M is a model of a HyperLTL formula φ, denoted by $M \models \varphi$, whenever $\mathsf{Traces}(M) \models \varphi$.

3 The Logic SC-HyperLTL for Finite Traces

We now introduce SC-HyperLTL, a decidable logic that combines the stuttering feature from Stuttering HyperLTL with a limited version of the context feature from Context HyperLTL. Our logic is expressive and at the same time enjoys a decidable model-checking problem. Since both HyperLTL$_S$ and HyperLTL$_C$ are undecidable so is their unrestricted combination (see [8]), so our logic can be seen as a (decidable) useful fragment of such a rich logic. This logics extends HyperLTL with two new features from HyperLTL$_S$ and HyperLTL$_C$: $\mathit{\Gamma}$ operator and the singleton operator $\langle x \rangle$. The singleton context operator evaluates a formula over a single trace variable x ignoring whether or not the other traces terminated. The $\mathit{\Gamma}$-operator is used to restrict the evaluation of formulae to observation points determined by $\mathit{\Gamma}$; then, the temporal operators are synchronously evaluated on these observation points.

The syntax of SC-HyperLTL is:

$$\begin{aligned}
&\alpha := \forall x.\alpha \mid \exists x.\alpha \mid \mathit{\Gamma}.\varphi \\
&\varphi := \langle x \rangle \beta[x] \mid a[x] \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi \,\mathbf{U}\, \varphi \mid \mathbf{Y}\varphi \mid \varphi \,\mathbf{S}\, \varphi
\end{aligned}$$

where $a \in V$, $\mathit{\Gamma}$ is a set of PLTL formulae over V, β is a PLTL formula over V and $\beta[x]$ is defined by replacing each variable $a \in V$ occurring in β with $a[x]$.

To define the semantics, we first introduce $\mathit{\Gamma}$-stutter factorizations, adapting the definition from [14].

Definition 1 ($\mathit{\Gamma}$-stutter factorization).
Let $\mathit{\Gamma}$ be a finite set of PLTL formulas and σ a finite trace. The $\mathit{\Gamma}$-stutter factorization of σ is the unique increasing sequence of positions $\{i_0 \dots i_k\}$ such that $i_0 = 0$, $i_k = |\sigma| - 1$ and:

- *for each $\theta \in \mathit{\Gamma}$ and j, $0 \leq j < k$, the truth value of θ along $[i_j, i_{j+1})$ does not change, i.e., for all $h, l \in [i_j, i_{j+1})$, $(\sigma, h) \models \theta$ if and only if $(\sigma, l) \models \theta$.*
- *the truth value of some formula in $\mathit{\Gamma}$ changes along adjacent segments (except at i_k), i.e., for each $0 \leq j < k - 1$, for some $\theta \in \mathit{\Gamma}$, $(\sigma, i_j) \models \theta$ if and only if $(\sigma, i_{j+1}) \not\models \theta$.*

In order to define the *relevant positions*, we define a finite version of the successor and predecessor relations. We use a special symbol `und` (meaning undefined) to denote the successor of the last point in the trace and the predecessor of the first point in the trace.

Definition 2 (Relativized successor).
Let $\mathit{\Gamma}$ be a finite set of PLTL formulas and σ a finite trace. The relativized successor and relativized predecessor are defined as follows:

$$succ_{\mathit{\Gamma}}(\sigma, i) := \begin{cases} \mathtt{und} & \text{If } i \geq |\sigma| - 1 \\ (\sigma, i_{j+1}) & \text{If } i \in [i_j, i_{j+1}) \text{ for some } j \in [0, k) \end{cases}$$

$$pred_{\mathit{\Gamma}}(\sigma, i) := \begin{cases} \mathtt{und} & \text{If } i = 0 \text{ or } i > |\sigma| - 1 \\ (\sigma, i_j) & \text{If } i \in (i_j, i_{j+1}] \text{ for some } j \in [0, k) \end{cases}$$

We extend the previous definitions to the trace assigment $\mathit{\Pi}$ as follows; $succ_{\mathit{\Gamma}}(\mathtt{und}) = \mathtt{und}$ if there is a trace variable $x \in \mathrm{VAR}$ such that the successor $succ_{\mathit{\Gamma}}(\mathit{\Pi})(x) = \mathtt{und}$. Similarly, $pred_{\mathit{\Gamma}}(\mathit{\Pi}) = \mathtt{und}$ if there is a trace variable

x such that the predecessor $pred_{\Gamma}(\Pi)(x) = \texttt{und}$. Otherwise, $succ_{\Gamma}(\Pi)(x) = succ_{\Gamma}(\Pi(x))$ and $pred_{\Gamma}(\Pi)(x) = pred_{\Gamma}(\Pi(x))$ for each $x \in \mathrm{VAR}$.

Finally, we define the semantics of SC-HyperLTL as follows:

$$\begin{array}{rl}
\Pi \models_T \exists x.\alpha & \Leftrightarrow \text{for some } \sigma \in T,\ \Pi[x \mapsto (\sigma, 0)] \models_T \alpha \\
\Pi \models_T \forall x.\alpha & \Leftrightarrow \text{for all } \sigma \in T,\ \Pi[x \mapsto (\sigma, 0)] \models_T \alpha \\
\Pi \models_T \Gamma.\varphi & \Leftrightarrow (\Pi, \Gamma) \models \varphi \\
(\Pi, \Gamma) \models a[x] & \Leftrightarrow a \in \sigma(p),\ \text{where } (\sigma, p) = \Pi(x) \\
(\Pi, \Gamma) \models \varphi_1 \mathbf{U} \varphi_2 & \Leftrightarrow \text{for some } j \geq 0\ \ succ^j_{\Gamma}(\Pi) \neq \texttt{und}, (succ^j_{\Gamma}(\Pi), \Gamma) \models \varphi_2, \\
& \quad\ \text{and for all } i < j, (succ^i_{\Gamma}(\Pi), \Gamma) \models \varphi_1 \\
(\Pi, \Gamma) \models \mathbf{X}\varphi & \Leftrightarrow succ_{\Gamma}(\Pi) \neq \texttt{und} \text{ and } (succ_{\Gamma}(\Pi), \Gamma) \models \varphi \\
(\Pi, \Gamma) \models \varphi_1 \mathbf{S} \varphi_2 & \Leftrightarrow \text{for some } j \geq 0\ \ pred^j_{\Gamma}(\Pi) \neq \texttt{und}, (pred^j_{\Gamma}(\Pi), \Gamma) \models \varphi_2, \\
& \quad\ \text{and for all } i < j, (pred^i_{\Gamma}(\Pi), \Gamma) \models \varphi_1 \\
(\Pi, \Gamma) \models \mathbf{Y}\varphi & \Leftrightarrow pred_{\Gamma}(\Pi) \neq \texttt{und} \text{ and } (pred_{\Gamma}(\Pi), \Gamma) \models \varphi \\
(\Pi, \Gamma) \models \langle x \rangle \beta[x] & \Leftrightarrow (\sigma, i) \models \beta \text{ with } (\sigma, i) = \Pi(x)
\end{array}$$

Theorem 1. *Model Checking of SC-HyperLTL is decidable.*

The proof proceeds by interpreting the logic on infinite traces, which is proven decidable in [8] via a reduction into QPTL (quantified propositional LTL) [36]. The resulting QPTL formula encodes the context and stuttering modalities using special padding letters and encoding the states of the model, and additionally requiring self-loop fresh states to create infinite traces from models with finite traces. The encoding from SC-HyperLTL to the logic defined in [8] can be found in the Appendix of the extended version of the manuscript [9].

Reversing formulas. We now introduce an interesting property of our logic on finite traces. We first define the reverse function on formulas.

Definition 3 (Reverse SC-HyperLTL formula). *Let φ be a SC-HyperLTL formula. We define $rev(\varphi)$ as follows (here, $\Gamma_R := \{\mathbf{X}rev(\theta) | \theta \in \Gamma\}$):*

$$\begin{array}{rlrl}
rev(\forall x.\varphi) & := \forall x.rev(\varphi) & rev(\mathbf{X}\psi) & := \mathbf{Y}rev(\psi) \\
rev(\exists x.\varphi) & := \exists x.rev(\varphi) & rev(\mathbf{Y}\psi) & := \mathbf{X}rev(\psi) \\
rev(\Gamma.\psi) & := \Gamma_R.rev(\psi) & rev(\langle x \rangle \psi) & := \langle x \rangle rev(\psi) \\
rev(v[x]) & := v[x] & rev(\psi_1 \mathbf{U} \psi_2) & := rev(\psi_1) \mathbf{S}\, rev(\psi_2) \\
rev(\psi_1 \vee \psi_2) & := rev(\psi_1) \vee rev(\psi_2) & rev(\psi_1 \mathbf{S} \psi_2) & := rev(\psi_1) \mathbf{U}\, rev(\psi_2) \\
rev(\neg\psi) & := \neg rev(\psi) & &
\end{array}$$

The reverse of a formula considers the trace in the opposite time direction, which is enabled by having future and past operators and traces having a beginning and an end. Since Γ points are evaluated considering the points with different past evaluation, we need to reverse Γ as well. Given a trace σ we use σ^{-1} for the trace such that $\sigma^{-1}(i) = \sigma(N - i)$ where $N = |\sigma| - 1$. We use Π^{-1} for the trace assignment that assigns σ^{-1} to x when Π assigns σ to x. The following holds directly from the definition.

Theorem 2. *For every formula φ, $rev(rev(\varphi)) \equiv \varphi$. Given trace assignment Π, $(\Pi, \Gamma) \models \varphi$ if and only if $(\Pi^{-1}, \Gamma_R) \models rev(\varphi)$.*

Properties Expressible in SC-HyperLTL. We now show some important practical properties that can be expressed in the decidable fragment of the logic (others include observation determinism, initial-state opacity, etc).

GMNI[26]: Goguen and Meseguer non-interference states that the observations of public information do not change when the secret information is removed. We can define this property for the asynchronous setting with the following formula:

$$\forall x.\exists y.\{lo\}.\langle x\rangle\mathbf{G}\lambda[x] \wedge \mathbf{G}(lo[x] \leftrightarrow lo[y]),$$

where lo represents the public information and λ represents the absence of secret inputs.

Observational Determinism[38]: Observational determinism states that if two executions have the same initial low input, then the two executions will be indistinguishable observing only low output. In the asynchronous setting we can express it with the following property, where LI and LO are respectively the sets of low inputs and low outputs.

$$\forall x.\forall y.\{LO\}.\bigwedge_{v\in LI}(v[x] \leftrightarrow v[y]) \rightarrow \bigwedge_{v\in LO}\mathbf{G}(v[x] \leftrightarrow v[y]),$$

Battery Sensor. Consider the high level model shown in Fig. 1, which describes a battery sensor. To deal with hardware faults in the sensor, a second sensor is used as a backup, and a Selector component chooses which sensor output (sensed1/sensed2) shall be considered. The selector component is connected with the Recovery component which detects faults like abnormal oscillations and triggers a recovery action by sending a "recovery" signal (dashed blue line). The Selector component will then choose Sensor2 as the main sensor. We assume that faults are permanent and that sensors run asynchronously.

For any execution with a fault, all observationally equivalent executions (from the Recovery component perspective) trigger a recovery command within a bounded amount of steps, where an observation occurs when *sensed* changes:

$$\varphi_{rec} := \forall x.\forall y.\{\mathit{sensed}\}.\mathbf{G}(\mathit{obseq} \wedge \langle x\rangle \mathit{faulted}[x] \rightarrow \langle y\rangle\mathbf{F}^{\leq d}\mathit{rcv}[y])$$

where $\mathbf{F}^{\leq d}\phi$ is the bounded future operator defined as $\phi \vee \mathbf{X}(\mathbf{F}^{\leq d-1}\phi)$ with $\mathbf{F}^{\leq 0} := \phi$, $\mathit{obseq} := \mathbf{H}(\mathit{sensed}[x] \leftrightarrow \mathit{sensed}[y])$, $\mathit{faulted} := (\neg \mathit{chd(sensed)}\,\mathbf{S}\,\mathit{fault})$ and *chd(sensed)* determines whether the sensed value changed.

The set $\{\mathit{sensed}\}$ is used to consider only the relevant points in which the sensed value changes. The formula *obseq* ensures that the two traces have the same sequences of sensed values. Then, $\mathit{faulted}[x]$ states that a fault occurred between the last observation point and the current one. Finally, the consequence is that the recovery command is activated within d steps in the related trace y.

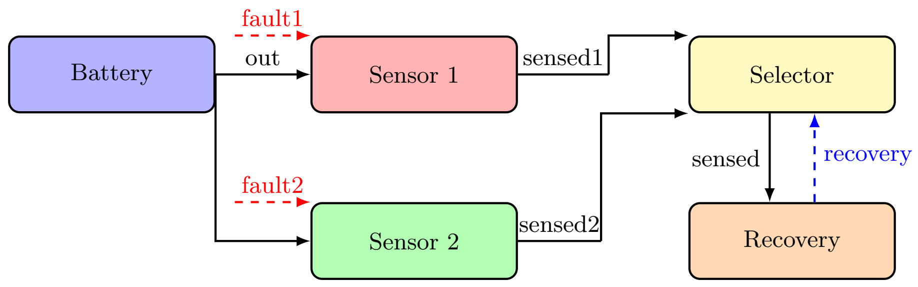

Fig. 1: Redundant Sensor System with Fault Detection, Isolation, and Recovery component schema. The coloured rectangles represents the internal components, the black arrow represent the data-port connection between components, the red dashed arrows represent fault events and, the blue dashed arrow represent the recovery signal.

4 Model checking SC-HyperLTL

We now introduce two algorithms to verify SC-HyperLTL. These methods are defined for the finite-trace semantics studied in this paper, but they can be easily adapted to infinite-trace semantics.

4.1 Algorithm 1: Stuttering Encoding

The main idea is to extend the transition system to encode stuttering and introduce sub-formulas that encode alignment constraints for the observation points. The resulting transition system and formula can be verified using standard HyperLTL model checking. The model transformation is as follows.

Definition 4 (Stuttered extension). *Let M be a STS. We define the stuttered extension of M as $M^{st} := \langle V \cup \{st, halt\}, I \wedge \neg halt, T^{st}\rangle$ where st, halt are two fresh Boolean variables and $T^{st} = ((st \vee halt) \to \bigwedge_{v\in V}(v' = v)) \wedge ((\neg st \wedge \neg halt) \to T) \wedge (halt \to halt')$ modifies T including stuttering transitions and halting self-loops.*

Essentially, M^{st} duplicates every state of M with a new version that models stuttering. M^{st} also adds a halting self-loop for every final state, so every accepting word of M is extended into an infinite word in M^{st}.

Definition 5 (Alignment under stuttering). *Let Γ be a finite set of PLTL formulae, $x_1, \ldots, x_n$ be a collection of trace variables. We define θ_Γ^{st} as follows:*

$$\theta_\Gamma^{st} := \mathbf{G}(\neg end^{st} \to \bigwedge_{1<i\leq n} (\theta_\Gamma[x_i] \leftrightarrow \theta_\Gamma[x_1])), \qquad \textit{where}$$

– $\theta_\Gamma := \mathit{first}^{st} \vee \mathit{last}^{st} \vee \bigvee_{\theta\in\Gamma}(\mathbf{Y}\theta \leftrightarrow \neg\theta)$ *represents the observation points,*

– *$end^{st} := \bigvee_{1 \leq i \leq n} halt[x_i]$ models the termination of some trace,*
– *$first^{st} := \mathbf{Z}\bot$ captures the first state of a trace (the same for all traces), and*
– *$last^{st} := \neg halt \wedge \mathbf{X} halt$ represents the last state.*

Def. 5 introduces the alignment constraints of the traces at the observation points. The alignment is preserved up to the "end" of the "shortest" stuttered trace. The formula θ_Γ captures observation points, which include the initial state $first^{st}$ and the final state of the trace $last^{st}$. The antecedent of the implication holds whenever one of the traces has not surpassed the final state yet, while the consequent is the same truth value of observations of each trace. This constraint synchronously evaluates the observation points by padding the non-aligned observations using stuttering expansion.

We now introduce a variation of rewriting of formulas from [10] adapted to SC-HyperLTL. This rewriting $\mathcal{R}$ takes two quantifier-free and Γ-free SC-HyperLTL formulae ψ and φ_{st} and produces a quantifier-free HyperLTL formula:

Definition 6 (Stuttering Rewriting of SC-HyperLTL). *Let φ be a quantifier free SC-HyperLTL formula without Γ operator. We define the rewriting $\mathcal{R}_{\varphi_{st}}(\psi)$ as follows:*

$$\begin{aligned}
\mathcal{R}_{\varphi_{st}}(v[x]) &:= v[x] \\
\mathcal{R}_{\varphi_{st}}(\neg\psi) &:= \neg\mathcal{R}_{\varphi_{st}}(\psi) \\
\mathcal{R}_{\varphi_{st}}(\psi_1 \vee \psi_2) &:= \mathcal{R}_{\varphi_{st}}(\psi_1) \vee \mathcal{R}_{\varphi_{st}}(\psi_2) \\
\mathcal{R}_{\varphi_{st}}(\langle x \rangle \psi) &:= \mathcal{R}_{st[x] \vee halt[x]}(\psi) \\
\mathcal{R}_{\varphi_{st}}(\mathbf{X}\psi) &:= \mathbf{X}(\varphi_{st}\, \mathbf{U}\, (\neg\varphi_{st} \wedge \mathcal{R}_{\varphi_{st}}(\psi))) \\
\mathcal{R}_{\varphi_{st}}(\mathbf{Y}\psi) &:= \mathbf{Y}(\varphi_{st}\, \mathbf{S}\, (\neg\varphi_{st} \wedge \mathcal{R}_{\varphi_{st}}(\psi))) \\
\mathcal{R}_{\varphi_{st}}(\psi_1\, \mathbf{U}\, \psi_2) &:= (\varphi_{st} \vee \mathcal{R}_{\varphi_{st}}(\psi_1))\, \mathbf{U}\, (\neg\varphi_{st} \wedge \mathcal{R}_{\varphi_{st}}(\psi_2)) \\
\mathcal{R}_{\varphi_{st}}(\psi_1\, \mathbf{S}\, \psi_2) &:= (\varphi_{st} \vee \mathcal{R}_{\varphi_{st}}(\psi_1))\, \mathbf{S}\, (\neg\varphi_{st} \wedge \mathcal{R}_{\varphi_{st}}(\psi_2))
\end{aligned}$$

Finally, let $\varphi := \Gamma.\psi$ and let x_1 be a trace variables occuring in ψ. We denote $\mathcal{R}(\varphi) := \mathcal{R}_{\neg\theta_\Gamma[x_1]}$.

Temporal operator formulae are evaluated at positions where φ_{st} is false (that is, aligned at non-stuttering positions). For example, $\mathcal{R}_{\varphi_{st}}(\mathbf{X}p[x])$ would be true under the HyperLTL semantics only if at the next occurrence of $\neg\varphi_{st}$ the variable $p[x]$ is true. A detailed description of the PLTL part of the rewriting can be found in [10].

For singleton-context formulas, φ_{st} is replaced for the subformula with $st[x] \vee halt[x]$ that represents either a local stuttering state or the end of the local trace. The intuition is that, with singleton context, we want to only evaluate the points that are relevant for the local trace. For non-singleton formulae φ_{st} is set to $\neg\theta_\Gamma[x_1] \vee end^{st}$. Therefore, each temporal operator is evaluated only at the observation points of x_1 in which no trace has terminated yet. Provided the alignment of Def. 5, the observation points that need to be evaluated with finite semantics are the ones in which $\theta_\Gamma[x_1] \wedge \neg end^{st}$ is satisfied.

Theorem 3. *Let M be a STS and ψ be a quantifier-free SC-HyperLTL formula. Then,*

$$\begin{array}{lll} M \models \forall x_1 \ldots \forall x_n . \mathit{\Gamma} . \psi & \textit{if and only if} & M^{st} \models \forall x_1 \ldots \forall x_n . \theta^{st}_{\mathit{\Gamma}} \to \mathcal{R}(\psi) \\ M \models \exists x_1 \ldots \exists x_n . \mathit{\Gamma} . \psi & \textit{if and only if} & M^{st} \models \exists x_1 \ldots \exists x_n . \theta^{st}_{\mathit{\Gamma}} \wedge \mathcal{R}(\psi) \end{array}$$

Proof. (Sketch) The constraint $\theta^{st}_{\mathit{\Gamma}}$ forces the traces to be aligned at the positions in which one of the elements of $\mathit{\Gamma}$ changes or at the final or initial positions. Given this alignment, $\mathcal{R}$ parses $\mathit{\Gamma}$-temporal operators evaluating them only in the $\mathit{\Gamma}$ positions of the first trace (which is aligned with the other traces). Note that in SC-HyperLTL context operators occur only with singleton contexts and with sub-formulae over the context trace, so $\mathcal{R}_{\varphi_{st}}(\langle x \rangle \psi)$ restricts the evaluation of ψ to the points in which $st[x] \vee halt[x]$ is false, that is, on the local points of the trace. It is easy to see that the original semantics is preserved by the translation. The complete proof is in the Appendix of the extended version of the manuscript [9].

4.2 Algorithm 2: Bounded Observations

We now introduce an alternative algorithm to deal with asynchrony for formulas with arbitrary quantifier alternations. This algorithm introduces k history variables $\{v_1, \ldots v_k\}$ for each variable v occurring inside the formula. The intended meaning of v_j is to mimic the value that v has at the j-th position of the stuttering factorization of the trace. Then, the variables are used in a BMC-like approach [7] to check whether the property holds on a horizon with at most k observations. If k observations are sufficient to prove or disprove the property then the outcome is informative. Note that the bound k is on the number of observations and not on the length of the trace, which is unbounded.

To simplify the encoding we consider formulae without singleton context expressions. Every single context sub-formula can be translated into automata with standard techniques, and the sub-formula can be replaced by a fresh variable as shown in Prop. 1 below. In this section, given an LTL$_f$ formula β over variables V', we denote M_β as the STS with alphabet $V \cup \{v_\beta\}$ such that for each (finite) pointed trace (σ, i), $(\sigma, i) \models \beta \Leftrightarrow (\sigma, i) \models_f v_\beta$. We use $Sub(\varphi)$ for the set of all the sub-formulas of φ.

Proposition 1 (Removing singleton context) *Let M be a STS and α be a SC-HyperLTL formula. $M \models_f \alpha$ if and only if $(M \times \bigtimes_{\langle x_i \rangle \beta[x_i] \in Sub(\varphi)} M_\beta) \models_f \alpha'$, where α' is obtained by replacing each occurrence of $\langle x_i \rangle \beta[x_i]$ with $v_\beta[x]$.*

Definition 7 (Padding traces). *Let M be a STS, we define M^{pad} as the STS with a self-loop transition at the end of each execution: $M^{pad} := \langle V \cup \{end\}, I \wedge \neg end, (\neg end' \to T) \wedge (end' \to \bigwedge_{v \in V} v = v') \wedge (end \to end'), F \cup \{end\} \rangle$*

To deal with the fact that local traces might have different lengths, Def. 7 introduces the notion of padding traces by using the *end* variable.

Definition 8 (History-variable STS). *Let M be a STS, $\overline{V} \subseteq V$ be a set of variables, $\overline{V}^k := \{v_i \mid v \in \overline{V}, 0 \leq i < k\}$ be the set containing k copies of the variables of $\overline{V}$ and pos be an enum variable ranging from 0 to $k+1$.*

We define the STS $P^k_{\theta_\Gamma,\overline{V}} := \langle V \cup \overline{V}^k \cup \{pos, v_{\theta_\Gamma}, end\}, pos = 0, T^k\rangle$ with

$$T^k := \left(\begin{array}{c} (obs_\Gamma \to pos' = pos + 1) \wedge (\neg obs_\Gamma \to pos' = pos) \wedge \\ \bigwedge_{v_i \in \overline{V}^k} ((pos = i \wedge obs_\Gamma \to v'_i = v) \wedge (pos \neq i \vee \neg obs_\Gamma \to v'_i = v_i)) \end{array} \right)$$

where v_{θ_Γ} represents the points in which elements of Γ changed, $obs_\Gamma := \neg end \wedge (v_{\theta_\Gamma} \vee end' \vee pos = 0)$ represents the observation points in the trace i.e. either the points in which elements of Γ change, the initial state or the last point of the trace.

Def. 8 introduces the behaviour of history variables. Variable *pos* counts the observations seen so far, initialized to 0, and increased each time an observation is encountered. If, more than k observation are encountered *pos* takes the value $k+1$ (encoding that more than k observations have happened). Each variable v_i mimics the value of v at the i-th observation. When $pos = i$, it takes the current value of v ($v'_i = v$), otherwise it stutters ($v'_i = v_i$). It should be noted that, before the i-th observation, v_i can take any value. Variable v_{θ_Γ} represents the points of observations, where one of the $\theta \in \Gamma$ changes. Fig. 2 shows two traces of a system, with $\Gamma = \{a\}$ and $k = 3$.

Definition 9 (K-bound unrolling). *We define the bounded unrolling of quantifier free SC-HyperLTL formulas up to k. The complete encoding assumes formulae in negation normal form i.e. in which negations only occur over predicates.*

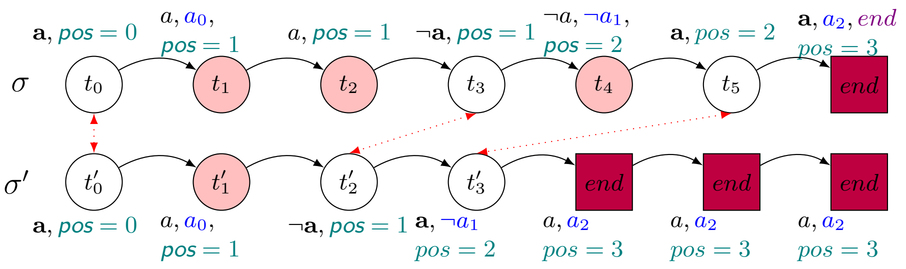

Fig. 2: Traces σ, σ' with the construction in Alg. 2 for $k = 3$ and $\Gamma = \{a\}$. History variables are shown in blue; the end of the local traces are encoded by *end* variables, shown in purple; *pos* variables are shown in teal. Red dotted lines map the Γ-points of σ with the corresponding Γ-points of σ'. Pink circles represent the non-interesting trace points and purple boxes the end of the traces. The trace σ' is shorter than σ, so σ' is extended to be aligned with σ in the very last position according to Def. 7.

${}^{\rho}[\![\varphi]\!]_k^k := \rho$. *For all* $i < k$:

$$
\begin{array}{rlrl}
{}^{\rho}[\![v[x]]\!]_i^k &:= v_i[x] & {}^{\rho}[\![\neg v[x]]\!]_i^k &:= \neg v_i[x] \\
{}^{\rho}[\![\psi_1 \lor \psi_2]\!]_i^k &:= {}^{\rho}[\![\psi_1]\!]_i^k \lor {}^{\rho}[\![\psi_2]\!]_i^k & {}^{\rho}[\![\psi_1 \land \psi_2]\!]_i^k &:= {}^{\rho}[\![\psi_1]\!]_i^k \land {}^{\rho}[\![\psi_2]\!]_i^k \\
{}^{\rho}[\![\mathbf{X}_\Gamma \psi]\!]_i^k &:= \neg end_\Gamma^{i+1} \land {}^{\rho}[\![\psi]\!]_{i+1}^k & {}^{\rho}[\![\mathbf{N}_\Gamma \psi]\!]_i^k &:= end_\Gamma^{i+1} \lor {}^{\rho}[\![\psi]\!]_{i+1}^k
\end{array}
$$

$$
\begin{aligned}
{}^{\rho}[\![\psi_1\, \mathbf{U}_\Gamma\, \psi_2]\!]_i^k &:= {}^{\rho}[\![\psi_2]\!]_i^k \lor ({}^{\rho}[\![\psi_1]\!]_i^k \land {}^{\rho}[\![\mathbf{X}\top]\!]_{i+1}^k \land {}^{\rho}[\![\psi_1\, \mathbf{U}_\Gamma\, \psi_2]\!]_{i+1}^k) \\
{}^{\rho}[\![\psi_1\, \mathbf{R}_\Gamma\, \psi_2]\!]_i^k &:= {}^{\rho}[\![\psi_2]\!]_i^k \land ({}^{\rho}[\![\psi_1]\!]_i^k \lor {}^{\rho}[\![\mathbf{N}\bot]\!]_{i+1}^k \lor {}^{\rho}[\![\psi_1\, \mathbf{R}_\Gamma\, \psi_2]\!]_{i+1}^k) \\
{}^{\rho}[\![\mathbf{Y}_\Gamma \psi]\!]_0^k &:= \bot \quad {}^{\rho}[\![\mathbf{Z}_\Gamma \psi]\!]_0^k := \top \quad {}^{\rho}[\![\mathbf{Y}_\Gamma \psi]\!]_i^k := {}^{\rho}[\![\psi]\!]_{i-1}^k \quad {}^{\rho}[\![\mathbf{Z}_\Gamma \psi]\!]_i^k := {}^{\rho}[\![\psi]\!]_{i-1}^k \\
{}^{\rho}[\![\psi_1\, \mathbf{S}_\Gamma\, \psi_2]\!]_i^k &:= {}^{\rho}[\![\psi_2]\!]_i^k \lor ({}^{\rho}[\![\psi_1]\!]_i^k \land {}^{\rho}[\![\mathbf{Y}\top]\!]_i^k \land {}^{\rho}[\![\psi_1\, \mathbf{S}_\Gamma\, \psi_2]\!]_{i-1}^k) \\
{}^{\rho}[\![\psi_1\, \mathbf{T}_\Gamma\, \psi_2]\!]_i^k &:= {}^{\rho}[\![\psi_2]\!]_i^k \land ({}^{\rho}[\![\psi_1]\!]_i^k \lor ({}^{\rho}[\![\mathbf{Z}\bot]\!]_i^k \lor {}^{\rho}[\![\psi_1\, \mathbf{T}_\Gamma\, \psi_2]\!]_{i-1}^k))
\end{aligned}
$$

where $end_\Gamma^i := \bigvee_{x \in VAR}(pos[x] \leq i)$, $\rho \in \{\bot, \top\}$.

The formula ${}^{\rho}[\![\psi]\!]_0^k$ introduces an unrolling of formulas following the finite semantics of the operators, using the history variables as if they were synchronous projections of each trace and then traverse the traces synchronously. Since each trace might have a different number of observations, the encoding considers the end of the projected trace as the position i such that at least one trace has $pos = i$ (end_Γ^i).

The following theorem establishes the correctness of the reduction from SC-HyperLTL to HyperLTL with finite semantics using Def. 7, Def. 8 and, Def. 9.

Theorem 4. *Let* M *be a STS, let* $\varphi := \mathcal{Q}_n x_n \ldots \mathcal{Q}_1 x_1 . \Gamma . \psi$ *be a SC-HyperLTL formula, where each* $\mathcal{Q}_i \in \{\forall, \exists\}$ *is a quantifier, let* $\theta_\Gamma := \bigvee_{\theta \in \Gamma}(\neg\theta \leftrightarrow \mathbf{Y}\theta)$. *Consider* $M_{\mathcal{B}}^k := M^{pad} \times M_{\theta_\Gamma} \times P_{\theta_\Gamma, \overline{V}}^k$ *to be the STS constructed by the synchronous composition of* $P_{\theta_\Gamma, \overline{V}}^k$ *with* M *and* M_{θ_Γ} *extended with end transitions. If* $M_{\mathcal{B}}^k \models_f \mathbf{G}(pos \leq k)$ *holds, then* $M \models_f \mathcal{Q}_n x_n \ldots \mathcal{Q}_1 x_1 . \psi$ *iff* $M_{\mathcal{B}}^k \models_f \mathcal{Q}_n x_n \ldots \mathcal{Q}_1 x_1 . \mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\bot}[\![\psi]\!]_0^k)$. *Moreover,*

- *If* $M_{\mathcal{B}}^k \models_f \mathcal{Q}_n x_n \ldots \mathcal{Q}_1 x_1 . \mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\bot}[\![\psi]\!]_0^k)$ *then* $M \models_f \mathcal{Q}_n x_n \ldots \mathcal{Q}_1 x_1 . \psi$.
- *If* $M_{\mathcal{B}}^k \not\models_f \mathcal{Q}_n x_n \ldots \mathcal{Q}_1 x_1 . \mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\top}[\![\psi]\!]_0^k)$ *then* $M \not\models_f \mathcal{Q}_n x_n \ldots \mathcal{Q}_1 x_1 . \psi$

The theorem establishes a correspondence between the model-checking of a SC-HyperLTL formula in the original STS M and in the synchronous composition of M^{pad}, M_{θ_Γ} and $P_{\theta_\Gamma, \overline{V}}^k$. The extended system $M_{\mathcal{B}}^k$ incorporates the evaluation of Γ points using M_{θ_Γ}. The extended system considers traces with possibly different lengths via M^{pad}, and incorporates $P_{\theta_\Gamma, \overline{V}}^k$ to keep track of the assignments of variables in the observational point. The first part of the theorem states that, if M has at most k observations, then the unrolling ${}^{\rho}[\![\psi]\!]_0^k$ is satisfied at the end of the traces (i.e. when $\mathbf{N}\bot$ is true) if and only if the original model-checking problem holds. The second part of the theorem considers a partial relation between the model checking results of both problems. If the unrolling provides a positive result then the original formula satisfies the property as well. If proving the property requires more than k steps, the rule ${}^{\rho}[\![\psi]\!]_k^k$ returns $\bot$ as result (with $\rho = \bot$). Conversely, with an "optimistic" view by setting $\rho = \top$, a negative result (i.e. the property is not satisfied) entails that M violates the original formula. This mimics the pessimistic (resp. optimistic) semantics [30] for synchronous BMC with the difference that in our algorithm the actual traces have unbounded length.

Beyond k-observations. It is possible to extend Alg. 2 relaxing the global limit of k-observations into k-difference-in-observations, where observations are stored and evaluated modulo k. In Alg. 2 the i-th observation is stored at point i (assuming $i < k$) and remains fixed for the rest of the trace. In the extended approach, the storage of observations would cycle (indexed by $i \mod k$), allowing properties to be checked dynamically over the trace. To support this, an additional constraint needs to keep track of which trace is the fastest, enabling evaluation relative to trace alignment. This modified algorithm can handle systems where the differences in observations is bounded by k.

5 Proof of Concept Implementation

We implemented the techniques described in Section 4 in a proof-of-concept implementation using the nuXmv model checker[16]. The implementation applies the translations described above and outputs either a self-composed SMV model (if the property has no quantifier alternation) or an AutoHyper model (see [6]) with its corresponding HyperLTL property.

To validate our approach, we use three examples: (1) the example in Section 1, (2) a variation of the battery sensor from Section 3 and (3) the parallel composition, under interleaving semantics, of two programs: *P0*, that can take an arbitrary number of steps to terminate, and *P1*, which terminates after k steps. A scheduler that decides non-deterministically which process runs: *P0* (if $sched = 0$) or *P1* (if $sched = 1$). We intend to prove that the execution of *P1* is *deterministic* no matter what *P0* does, encoded as follows:

$$\psi_{det} := \forall x.\forall y.\{P1.var\}.\langle x\rangle FSchd[x] \wedge \langle y\rangle FSchd[y] \rightarrow \mathbf{G}(P1.var[x] \leftrightarrow P1.var[y])$$

where $FSchd := \mathbf{G}(\mathbf{NN}\bot \rightarrow sched = 1)$ states that the last scheduled process is *P1*. The property states that, if in the last transition of both traces *P1* is scheduled, then the two programs have always the same value of variable *P1.var*. We analyzed variations with quantifier alternations, incorrect versions (with manual bugs), and versions with unbounded amount of observations and controlled scheduling for bounded observations. The experiments[5] were run on a cluster with Intel Xeon CPU 6226R nodes running at 2.9GHz with 32CPU, 12GB. The timeout for each run was one hour and the memory cap was set to 1GB with 1 CPU assigned for each instance. Table 1 reports the execution times of the approaches. We observe that nuXmv is able to solve quite rapidly all instances, but AutoHyper suffers a significant performance slowdown. One possible reason is that encoding the behavior as part of the formula has a significant impact

[5] All the data can be found at `https://es-static.fbk.eu/people/bombardelli/papers/spin25/spin25.tar.gz`

Table 1: Empirical evaluation, where $*$ means non-informative with k-bound due to too many observations (TO is 1h). Dashed line on tools mean that they cannot be solved with it. For instance q.alt. model cannot be checked with nuXmv and models with past (Battery Sensor) cannot be checked with AutoHyper.

Name	**Method**	**Bound**	**Outcome**	**nuXmv Time (s)**	**AutoHyper Time (s)**
Process $n = 2$	k-bound (Alg.2)	3	True	2.82	5.54
Process $n = 4$	k-bound (Alg.2)	5	True	4.03	TO
Process $n = 4$	Stuttering(Alg.1)	–	True	1.86	MO
Process $n = 6$	k-bound (Alg.2)	7	True	4.17	MO
Process $n = 6$	Stuttering(Alg.1)	–	True	2.65	MO
Process $n = 6$ (bug)	k-bound (Alg.2)	7	False	3.33	MO
Process $n = 6$ (bug)	Stuttering(Alg.1)	–	False	3.02	MO
Process q. alt. $n = 2$	k-bound (Alg.2)	3	True	–	34.93
Process q. alt. $n = 3$	k-bound (Alg.2)	4	True	–	TO
Motivating example	k-bound (Alg.2)	7	True	1245.95	TO
Motivating example	k-bound (Alg.2)	3	False*	34.64	TO
Motivating example	Stuttering(Alg.1)	–	True	20.88	6.19
Battery Sensor	k-bound (Alg.2)	7	False*	11.86	–
Battery Sensor	Stuttering(Alg.1)	–	True	6.03	–
Battery Sensor (bug)	Stuttering(Alg.1)	–	True	3.04	–

on explicit state model-checkers like AutoHyper. Another possibility is the additional variables introduced to deal with traces of different lengths. We leave the research of how to alleviate this problem as future work. More in general, we observe that Alg. 1, that applies stuttering rewriting (for quantifier-alternation free instances) is more successful (when applicable).

6 Conclusion

We introduced SC-HyperLTL, a novel temporal logic with past for expressing hyperproperties over finite traces, that combines features from two existing asynchronous hyperlogics: stuttering HyperLTL and context HyperLTL. SC-HyperLTL addresses the challenge of verifying systems with finite traces, where events occur asynchronously and past operators are required. We established the decidability of model-checking SC-HyperLTL and proposed two practical algorithms for its verification: one based on stuttering translation and the other that leverages auxiliary variables. We implement these in a proof-of-concept prototype based on the nuXmv model checker, showing the feasibility of the approach.

One major direction for future work is the verification of decidable extensions of our logic [8], which currently lacks practical verification techniques. Another future direction is to improve the performance. For Alg. 1, one possible direction is to try to reduce the problem to invariant checking. For Alg. 2, we will study how to relax the k-observation assumption. Finally, although the decidability proof provides an implicit upper bound on complexity, it would be interesting to determine the exact complexity of our logics. Additionally, it would be valuable to precisely characterize the complexity of both the stuttering and k-bound algorithms.

References

1. E. Bartocci, T. Ferrére, T. A. Henzinger, D. Nickovic, and A. O. da Costa. Flavors of sequential information flow. In *Proc. of the 23rd Int'l Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI'22)*, 2022.
2. E. Bartocci, T. A. Henzinger, D. Nickovic, and A. Oliveira da Costa. Hypernode Automata. In G. A. Pérez and J.-F. Raskin, editors, *34th International Conference on Concurrency Theory (CONCUR 2023)*, volume 279 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:16, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
3. J. Baumeister, N. Coenen, B. Bonakdarpour, B. Finkbeiner, and C. Sánchez. A temporal logic for asynchronous hyperproperties. In *Proc. of the 33rd Int'l Conf. on Computer Aided Verification (CAV'21), Part I*, volume 12759 of *LNCS*, pages 694–717. Springer, 2021.
4. R. Beutner. Automated software verification of hyperliveness. In B. Finkbeiner and L. Kovács, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 196–216, Cham, 2024. Springer Nature Switzerland.
5. R. Beutner and B. Finkbeiner. Software verification of hyperproperties beyond k-safety. In S. Shoham and Y. Vizel, editors, *Computer Aided Verification*, pages 341–362, Cham, 2022. Springer International Publishing.
6. R. Beutner and B. Finkbeiner. AutoHyper: Explicit-state model checking for HyperLTL. In *Proc. of the 29th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'23)*, volume 13993 of *LNCS*, pages 145–163. Springer, 2023.
7. A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proc. of the 5th Int'l Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS'99)*, volume 1579 of *LNCS*, pages 193–207. Springer, 1999.
8. A. Bombardelli, L. Bozzelli, C. Sánchez, and S. Tonetta. Unifying asynchronous logics for hyperproperties. *CoRR*, abs/2404.16778, 2024.
9. A. Bombardelli, L. Bozzelli, C. Sánchez, and S. Tonetta. (asynchronous) temporal logics for hyperproperties on finite traces - extended version. 2025. `https://es-static.fbk.eu/people/bombardelli/papers/spin25/spin25_ext.pdf`.
10. A. Bombardelli and S. Tonetta. Asynchronous composition of local interface LTL properties. In *Proc. of the 14th Int'l NASA Formal Methods Symposium (NFM'22)*, volume 13260 of *LNCS*, pages 508–526, 2022.
11. B. Bonakdarpour, P. Prabhakar, and C. Sánchez. Model checking timed hyperproperties in discrete-time systems. In *Proc. of the 12th NASA Formal Methods Symposium (NFM'2020)*, volume 12229 of *LNCS*, pages 311–328. Springer, 2020.
12. B. Bonakdarpour, C. Sánchez, and G. Schneider. Monitoring hyperproperties by combining static analysis and runtime verification. In *Proc. of the 8th Symp. on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'18)*, volume 11245 of *LNCS*, pages 8–27. Springer-Cham, 2018.
13. M. Bozzano, A. Cimatti, M. Gario, and S. Tonetta. Formal design of asynchronous fault detection and identification components using temporal epistemic logic. *Logical Methods in Computer Science*, Volume 11, Issue 4, Nov. 2015.
14. L. Bozzelli, A. Peron, and C. Sánchez. Asynchronous extensions of HyperLTL. In *Proc. of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'21)*, pages 1–13. IEEE, 2021.

15. L. Bozzelli, A. Peron, and C. Sánchez. Expressiveness and decidability of temporal logics for asynchronous hyperproperties. In *Proc. of the 33rd International Conference on Concurrency Theory (CONCUR'22)*, volume 243 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
16. R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta. The nuxmv symbolic model checker. In *International Conference on Computer Aided Verification*, 2014.
17. M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez. Temporal logics for hyperproperties. In *Proc. of the 3rd Conference on Principles of Security and Trust (POST 2014)*, volume 8414 of *LNCS*, pages 265–284. Springer, 2014.
18. M. R. Clarkson and F. B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.
19. N. Coenen, B. Finkbeiner, C. Hahn, and J. Hofmann. The hierarchy of hyperlogics. In *Proc. of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'19)*, pages 1–13. IEEE, 2019.
20. N. Coenen, B. Finkbeiner, C. Sánchez, and L. Tentrup. Verifying hyperliveness. In *Proc. of the 31st Int'l Conf. on Computer Aided Verification (CAV'19)*, volume 11561 of *LNCS*, pages 121–139. Springer, 2019.
21. A. Correnson, T. Nießen, B. Finkbeiner, and G. Weissenbacher. Finding $\forall\exists$ Hyperbugs using Symbolic Execution. *Proc. ACM Program. Lang.*, 8(OOPSLA2), Oct. 2024.
22. C. Eisner, D. Fisman, J. Havlicek, Y. Lustig, A. McIsaac, and D. V. Campenhout. Reasoning with temporal logic on truncated paths. In *Proc. of the 15th Int'l Conf. on Computer Aided Verification (CAV'03)*, volume 2725 of *LNCS*, pages 27–39. Springer, 2003.
23. B. Finkbeiner, M. Rabe, and C. Sánchez. Algorithms for model checking HyperLTL and HyperCTL*. In *In Proc. of the 27th Int'l Conf. on Computer Aided Verification (CAV'15)*, volume 9206 of *LNCS*, pages 30–48. Springer, 2015.
24. G. D. Giacomo, P. Felli, M. Montali, and G. Perelli. HyperLDLf: a logic for checking properties of finite traces process logs. In *Proc. of the 30th Int'l Joint Conf. on Artificial Intelligence (IJCAI'21)*, pages 1859–1865. ijcai.org, 2021.
25. G. D. Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Proc. of the 23rd Int'l Joint Conference on Artificial Intelligence (IJCAI'13)*, pages 854–860. IJCAI/AAAI, 2013.
26. J. A. Goguen and J. Meseguer. Security policies and security models. In *Proc. of the IEEE Symposium on Security and Privacy (S&P'82)*, pages 11–20. IEEE, 1982.
27. J. O. Gutsfeld, M. Müller-Olm, and C. Ohrem. Propositional dynamic logic for hyperproperties. In *Proc. of the 31st Int'l Conf. on Concurrency Theory (CONCUR'20)*, LIPIcs 171, pages 50:1–50:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
28. J. O. Gutsfeld, M. Müller-Olm, and C. Ohrem. Automata and fixpoints for asynchronous hyperproperties. *Proc. ACM Program. Lang.*, 5(POPL):1–29, 2021.
29. T.-H. Hsu, B. Bonakdarpour, B. Finkbeiner, and C. Sánchez. Bounded model checking for asynchronous hyperproperties. In S. Sankaranarayanan and N. Sharygina, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 29–46, Cham, 2023. Springer Nature Switzerland.

30. T.-H. Hsu, C. Sánchez, and B. Bonakdarpour. Bounded model checking for hyperproperties. In *Proc. of the 27th Int'l Conf on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'21). Part I*, volume 12651 of *LNCS*, pages 94–112. Springer, 2021.
31. O. Lichtenstein, A. Pnueli, and L. D. Zuck. The glory of the past. In *Logic of Programs*, volume 193 of *LNCS*, pages 196—218. Springer, 1985.
32. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag, 1992.
33. Z. Manna and A. Pnueli. *Temporal verification of reactive systems - safety.* Springer, 1995.
34. A. Pnueli. The temporal logic of programs. In *Proc. of the 18th IEEE Symp. on Foundations of Computer Science (FOCS'77)*, pages 46–67. IEEE CS Press, 1977.
35. M. N. Rabe. *A temporal logic approach to information-flow control.* PhD thesis, Saarland University, 2016.
36. A. P. Sistla, M. Y. Vardi, and P. Wolper. The Complementation Problem for Büchi Automata with Applications to Temporal Logic. *Theoretical Computer Science*, 49:217–237, 1987.
37. Y. Wang, M. Zarei, B. Bonakdarpour, and M. Pajic. Statistical verification of hyperproperties for cyber-physical systems. *ACM Transactions on Embedded Computing systems (TECS)*, 18(5s):92:1–92:23, 2019.
38. S. Zdancewic and A. C. Myers. Observational determinism for concurrent program security. In *Proc. of the 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, pages 29–43. IEEE, 2003.

A Proofs from Section 3

A.1 Decidability proof

Definition 10. *(Finite to infinite) Let φ be a SC-HyperLTL formula, we define a rewriting for SC-HyperLTL $\zeta^f(\varphi)$:*

$$\zeta^f(C, v[x]) := v[x] \qquad \zeta^f(C, \varphi_1 \vee \varphi_2) := \varphi_1^f \vee \varphi_2^f$$
$$\zeta^f(C, \neg\varphi) := \neg\varphi^f \qquad \zeta^f(C, \langle x\rangle\alpha[x]) := \langle x\rangle\zeta^f(\{x\}, \alpha[x])$$
$$\zeta^f(\mathbf{X}\varphi) := \mathbf{X}(\theta^f_{Dom(\Pi)} \wedge \varphi^f) \qquad \zeta^f(C, \varphi_1 \,\mathbf{U}\, \varphi_2) := \varphi_1^f \,\mathbf{U}\, (\theta^f_C \wedge \varphi_2^f)$$
$$\zeta^f(C, \mathbf{Y}\varphi) := \mathbf{Y}\varphi^f \qquad \zeta^f(C, \varphi_1 \,\mathbf{S}\, \varphi_2) := \varphi_1^f \,\mathbf{S}\, \varphi_2^f$$

where $\theta^f_C := \bigwedge_{x\in C}(\neg\mathbf{X}end[x])$ and $\varphi^f := \zeta^f(C, \varphi)$.

Finally, we denote the rewriting for quantified formulae ζ^f as

$$\zeta^f(\forall x.\varphi) := \forall x.\zeta^f(\varphi) \qquad \zeta^f(\exists x.\varphi) := \exists x.\zeta^f(\varphi)$$
$$\zeta^f(\Gamma.\varphi) := \Gamma^f.\zeta^f(VAR, \varphi)$$

where $\Gamma^f := \Gamma \cup \{\mathbf{X}end\}$

Proposition 1. *Let M be a STS and φ be a SC-HyperLTL formula, $M^{inf} := \mathcal{E}nd(M)$.*

$$M \models_f \varphi \Leftrightarrow M^{inf} \models \zeta^f(\varphi)$$

Proof. Consider Π as a trace assignment over $\mathsf{Traces}^{fin}(M)$ and Π^{end} as the trace assignment over $\mathsf{Traces}(M^{inf})$ as the corresponding assignment augmented with *end*. More formally, $\forall x \in \text{VAR} : \forall 0 \leq j < |\sigma|, \forall v \in V : \sigma, j \models_f v \Leftrightarrow \sigma^{inf} \models v$ where $(\sigma, i) = \Pi(x)$ and $(\sigma', i) = \Pi(x)$. We want to prove that $(\Pi, \Gamma) \models_f \psi \Leftrightarrow (\Pi^{end}, \Gamma^f) \models \zeta^f(\psi, Dom(\Pi))$ We prove the statement by induction on the structure of the formula.

Before starting the proof, we observe that (*)if $succ_\Gamma(\Pi) \neq \mathtt{und}$, $succ_{\Gamma^f}(\Pi^f)$ is the trace assignment augmented with end of $succ_\Gamma$. Moreover, (**)if $\Pi \neq \mathtt{und}$: $pred_\Gamma(\Pi) = pred_{\Gamma^f}(\Pi^{end})$ Finally, (***)$succ_\Gamma(\Pi) = \mathtt{und} \Leftrightarrow succ_{\Gamma^f} \not\models \theta^f_C$.

Base case: $\zeta^f(v[x], C)$: By hypothesis Π and Π^{end} have the same assignments over vars.

Inductive case:

- $(\neg, \vee)$: Follows by induction.
- $\langle x\rangle\alpha[x]$: This is proved by the reduction from finite to infinite of LTLf described in the literature.
- $\mathbf{X}$: $(\Pi, \Gamma) \models_f \mathbf{X}\psi \Leftrightarrow succ_\Gamma(\Pi) \neq \mathtt{und}$ and $succ_\Gamma(\Pi) \models_f \psi$. $succ_\Gamma(\Pi) = \mathtt{und}$ iff exists $x \in Dom(\Pi)$ .s.t. $(\sigma, |\sigma| - 1) = \Pi(x) \Leftrightarrow$ exists $x \in Dom(\Pi)$ s.t. $\sigma, j \models \mathbf{X}end$ with $(\sigma, j) = \Pi^{end}(x) \Leftrightarrow \Pi^{end} \not\models \theta^f_{Dom(\Pi)}$. Finally, by (*): $\Leftrightarrow \Pi^{end} \models \theta^f_C$ and $succ_{\Gamma^f}(\Pi^f) \models \zeta^f(\psi) \Leftrightarrow (\Pi^{end}, \Gamma) \models \zeta^f(\mathbf{X}, Dom(\Pi))$
- $\psi_1 \,\mathbf{U}\, \psi_2$: $(\Pi, \Gamma) \models_f \psi_1 \,\mathbf{U}\, \psi_2 \Leftrightarrow$ there exists $k \geq 0$ s.t. $succ^k_\Gamma(\Pi) \neq \mathtt{und}$, $(succ^k_\Gamma, \Gamma) \models_f \psi$ and for all $0 \leq j < k$: $(succ^j_\Gamma(\Pi), \Gamma) \models_f \psi_1$. By (*) and (***): $\Leftrightarrow$ there exists $k \geq 0$ s.t. $(succ^k_{\Gamma^f}(\Pi^f), \Gamma^f) \models \theta^f_{Dom(\Pi)}$ and $(succ^k_{\Gamma^f}(\Pi^f), \Gamma^f) \models \zeta^f(\psi_2, Dom(\Pi))$ and for all $0 \leq j < k$: $succ^j_{\Gamma^f}$.

- $\mathbf{Y}, \mathbf{S}$: Past is a simpler version of future.
- $\mathcal{OP}_{\{flip\}}$: Follows from the fact that in both cases the successor is the synchronous one.

Proof. From Proposition 1 we can rewrite finite semantics formula into infinite semantics formula. We can observe that, the obtained formula is in the decidable fragment of HyperLTL$_{SC}$ ([8]) proving the decidability of this logic.

A.2 Reversibility proof

Proposition 2. *For each trace σ, for all index i, ψ be an LTL$_f$ formula:*

$$\sigma, i \models_f \psi \Leftrightarrow \sigma^{-1}, |\sigma| - i - 1 \models_f rev(\psi)$$

Moreover, $\sigma, i \models_f \psi \Leftrightarrow \sigma, i \models_f rev(rev(\psi))$

Proof. We prove this proposition by induction on the structure of the formula. We denote with m_i as the corresponding index in σ^{-1} and formally defined as $m_i := |\sigma| - 1 - i$. Base case is trivial because we have only v. We consider the following trivial observations: $m_i = m_{i+1} - 1$, $i < j \Leftrightarrow m_i > m_j$, $m_0 = |\sigma| - 1$ and $m_{|\sigma|-1} = 0$.

- $\mathbf{X}\psi$: $\sigma, i \models_f \mathbf{X}\psi \Leftrightarrow i < |\sigma|-1$ and $\sigma, i+1 \models_f \psi \underset{ind.}{\Leftrightarrow} m_i > 0$ and $\sigma^{-1}, m_{i+1} \models_f rev(\psi) \underset{m_i = m_{i+1}+1}{\Leftrightarrow} \sigma^{-1}, m_i \models_f \mathbf{Y} rev(\psi) \Leftrightarrow \sigma^{-1}, m_i \models_f rev(\mathbf{X}\psi)$.
- $\psi_1 \mathbf{U} \psi_2$: $\sigma, i \models_f \psi_1 \mathbf{U} \psi_2 \Leftrightarrow$ there exists $|\sigma| - 1 \geq k \geq i$ s.t. $\sigma, k \models_f \psi_2$ and for all $i \leq j < k : \sigma, j \models_f \psi_1 \underset{ind.+rev}{\Leftrightarrow}$ there exists $0 \leq m_k \leq m_i$ s.t. $\sigma^{-1}, m_k \models_f rev(\psi_2)$ and for all $m_k < m_j \leq m_i : \sigma^{-1}, m_j \models_f \psi_1 \Leftrightarrow \sigma^{-1}, m_i \models_f rev(\psi_1) \, \mathbf{S} \, rev(\psi_2) \Leftrightarrow \sigma^{-1}, m_i \models_f rev(\psi_1 \, \mathbf{U} \, \psi_2)$.
- $\mathbf{Y}\psi$: $\sigma, i \models_f \mathbf{Y}\psi \Leftrightarrow i > 0$ and $\sigma, i-1 \models_f \psi \underset{ind.}{\Leftrightarrow} m_i < |\sigma|-1$ and $\sigma^{-1}, m_{i-1} \models_f rev(\psi) \underset{m_i = m_{i-1}-1}{\Leftrightarrow} \sigma^{-1}, m_i \models_f \mathbf{X} rev(\psi) \Leftrightarrow \sigma^{-1}, m_i \models_f rev(\mathbf{Y}\psi)$.
- $\psi_1 \mathbf{S} \psi_2$: $\sigma, i \models_f \psi_1 \mathbf{U} \psi_2 \Leftrightarrow$ there exists $0 \leq k \leq i$ s.t. $\sigma, k \models_f \psi_2$ and for all $i \geq j > k : \sigma, j \models_f \psi_1 \underset{ind.+rev}{\Leftrightarrow}$ there exists $m_i \leq m_k \leq |\sigma| - 1$ s.t. $\sigma^{-1}, m_k \models_f rev(\psi_2)$ and for all $m_k > m_j \geq m_i : \sigma^{-1}, m_j \models_f \psi_1 \Leftrightarrow \sigma^{-1}, m_i \models_f rev(\psi_1) \mathbf{U} \, rev(\psi_2) \Leftrightarrow \sigma^{-1}, m_i \models_f rev(\psi_1 \, \mathbf{S} \, \psi_2)$.

Finally, since $(\sigma^{-}1)^{-}1 = \sigma$ and $m_{m_i} = i$ it follows that $\sigma, i \models_f \psi \Leftrightarrow \sigma, i \models_f rev(rev(\psi))$.

Lemma 1 *Let $\mathcal{I}_\Gamma^\sigma$ be the stuttering sequence of the trace σ over the list of LTL formulae Γ:*

$$\mathcal{I}_\Gamma^\sigma := i_0 \dots i_k$$

Moreover, let the reverse of a sequence $\mathcal{I}$ of positions over a trace σ be defined as

$$[\mathcal{I}]^{-1} := \begin{cases} \overline{i_n}[\mathcal{I}']^{-1} & \textit{If } \mathcal{I} = \mathcal{I}' i_n \\ [i]^{-1} := \overline{i} & \textit{Otherwise} \end{cases}$$

where $\overline{i} := |\sigma| - i - 1$. *Then,*

$$[\mathcal{I}_{\Gamma}^{\sigma}]^0 = \mathcal{I}_{\Gamma_R}^{\sigma^{-1}}$$

Proof. By $succ_\Gamma$ and $pred_\Gamma$ definition, we have the following properties:

1. $i_0 = 0$, $i_k = |\sigma| - 1$ and $\forall j \in [0, k) : i_j < i_{j+1}$
2. For each $\theta \in \Gamma$, for all $j \in [0, k)$, for all $h \in [i_j, i_{j+1}) : \sigma, h \models_f \theta \Leftrightarrow \sigma, i_j \models_f \theta$
3. For all $j \in [0, k)$ there exists $\theta \in \Gamma$ s.t. $\sigma, i_j \models_f \theta \Leftrightarrow \sigma, i_{j+1} \not\models_f \theta$

From 1 it follows that $\overline{i_0} = |\sigma^{-1}| - 1$. Moreover, $i_k = |\sigma| - 1$ $\overline{i_k} = 0$. Moreover, since each $i_j < i_{j+1}$ we obtain with trivial arithmetic that $j \in [0, k) : \overline{i_{j+1}} < \overline{i_j}$. From which we derive thatthe reversed sequence satisfies 1).

From 2: For each $\theta \in \Gamma$, for all $j \in [0, k)$, for all $h \in [i_j, i_{j+1}) : \sigma, h \models_f \theta \Leftrightarrow \sigma, i_j \models_f \theta \Leftrightarrow$ For each $\theta \in \Gamma$, for all $j \in [0, k)$ for all $h \in (i_j, i_{j+1}) : \sigma, h \models_f \mathbf{Y}\theta \leftrightarrow \theta \underset{\text{Prop. 2}}{\Leftrightarrow} \ldots \sigma^{-1}, \overline{h} \models_f rev(\mathbf{Y}\theta \leftrightarrow \theta) \Leftrightarrow \ldots \sigma^{-1}, \overline{h} \models_f \mathbf{X}rev(\theta) \leftrightarrow rev(\theta)$. Since $0 < \overline{h} < |\sigma| - 1$, we can take $\theta' := \mathbf{X}rev(\theta)$ and derive: $\cdots \Leftrightarrow$ For each $\theta' \in \Gamma^{rev}$, for all $j \in [0, k)$, for all $h \in (i_j, i_{j+1}) : \sigma^{-1}, \overline{h} \models_f \mathbf{Y}\theta' \leftrightarrow \theta'$. Since h is universally quantified over (i_j, i_{j+1}) there is an $h = i_{j+1} - 1$ s.t. $\sigma^{-1}, \overline{i_{j+1} - 1} \models_f \mathbf{Y}\theta' \leftrightarrow \theta' \Leftrightarrow \sigma^{-1}, \overline{i_{j+1}} + 1 \models_f \mathbf{Y}\theta' \leftrightarrow \theta' \Leftrightarrow$ For each $\theta' \in \Gamma^{rev}$, for all $j \in [0, m_\infty)$ For all $h \in (i_j, i_{j+1})$: $\sigma^{-1}, \overline{h} \models_f \theta' \Leftrightarrow \sigma^{-1}, \overline{i_{j+1}} \models_f \theta'$. Therefore, we can conclude that 2 holds for the reverse sequence with Γ^{rev}. Note that it holds even if $i_k \neq |\sigma| - 1$ because in that case, while $\sigma^{-1}, 0 \models_f \theta \Leftrightarrow \sigma^{-1}, \overline{i_k} \models_f \theta$, for $\mathbf{X}rev(\theta)$ it holds the opposite; therefore, making the sequence starting from $0, \overline{i_k} \ldots$ satisfying property 2.

From 3: For all $j \in [0, k)$ there exists $\theta \in \Gamma$ s.t. $\sigma, i_j \models_f \theta \Leftrightarrow \sigma, i_{j+1} \not\models_f \theta$. Since by 2 each position outside of the sequence has the same truth value for Γ, we obtain: $\cdots \Leftrightarrow$ For all $\theta \in \Gamma$, for each $j \in (0, k] : \sigma, i_j \models_f \mathbf{Y}\theta \leftrightarrow \neg\theta \underset{rev}{\Leftrightarrow} \ldots \sigma^{-1}, \overline{i_j} \models_f rev(\mathbf{Y}\theta \leftrightarrow \neg\theta)$. We apply the same transformation applied for point 2. From which we derive: $\cdots \Leftrightarrow$ for all $\theta' \in \Gamma^{rev}$, for each $\overline{j \in [0, k)}$: $\sigma, \overline{i_j} \models_f \mathbf{Y}\theta' \leftrightarrow \neg\theta'$. Since 2 holds for our reverse sequence and since $\overline{k+1} = \overline{k} - 1$ for each k. Then, $\cdots \Leftrightarrow$ For all $\theta' \in \Gamma^{rev}$, for all $j \in [0, k) : \sigma^{-1}, \overline{i_j} \models_f \theta' \Leftrightarrow \sigma^{-1}, \overline{i_{j+1}} \models_f \neg\theta'$.

Proof of part one of theorem 2: To prove this part of the theorem we only need to prove that the stuttering factorization of $(\Gamma_R)^R$ is equivalent to the stuttering factorization of Γ since the temporal operator part of the theorem trivially follows from Prop. 2.

Since θ are LTL formulas double reverse can be removed, obtaining $(\Gamma_R)^R := \mathbf{X}rev(\mathbf{X}rev(\theta))|\theta \in \Gamma = \{\mathbf{XY}\theta | \theta \in \Gamma\}$. By stuttering factorization, each i_j is considered iff one of the $\theta \in \Gamma$ changed for the j-th time or for, $j = 0$ and $j = k$ if it is respectively initial or final state. Since θ is evaluated in the points $0 < j < k$, i.e. not at the "border" of the trace, then $\theta \equiv \mathbf{XY}\theta$ proving the equivalence.

From which we prove the first part of the theorem.

Proposition 2 $[succ_\Gamma(\Pi)]^{-1} = pred_\Gamma(\Pi^{-1})$ *and*
$[pred_\Gamma(\Pi)]^{-1} = succ_\Gamma(\Pi^{-1})$

Proof. Trivially follows from Lemma 1.

Proof of part 2 of Theorem 2 We denote in this proof $succ^i_{\Gamma}(\Pi) := \Pi_i$, $pred^i_{\Gamma}(\Pi) := \Pi_{-i}$, $succ^i_{\Gamma_R}(\Pi^{-1}) := \Pi^{-1}_i$ and $pred^i_{\Gamma_R}(\Pi^{-1}) := \Pi^{-1}_{-i}$.

We prove the theorem inductively on the structure of the formula.

The base case trivially holds since Π and Π^{-1} point to the same assignments.

- $(\neg, \vee)$ induction.
- $(\langle x\rangle\beta[x])$ From Prop. 2.
- **X**: $(\Pi, \Gamma) \models_f \mathbf{X}\psi \Leftrightarrow \Pi_1 \neq \texttt{und}$ and $(\Pi_1, \Gamma) \models_f \psi \underset{Prop\ 2}{\Leftrightarrow} (\Pi^{-1}_{-1} \neq \texttt{und}$ and $(\Pi^{-1}_{-1}, \Gamma_R) \models_f rev(\psi) \Leftrightarrow (\Pi^{-1}, \Gamma_R) \models_f \mathbf{Y} rev(\psi)$.
- (**U**) $(\Pi, \Gamma) \models_f \psi_1 \,\mathbf{U}\, \psi_2 \Leftrightarrow$ there exists $k \geq 0$ s.t. $\Pi_k \neq \texttt{und}$, $(\Pi_k, \Gamma) \models_f \psi_2$ and for all $0 \leq j < k : (\Pi_j, \Gamma) \models_f \psi_1 \underset{Prop\ 2}{\Leftrightarrow}$ there exists $k \geq 0$ s.t. $\Pi^{-1}_{-k} \neq \texttt{und}$, $(\Pi^{-1}_{-k}, \Gamma_R) \models_f rev(\psi_2)$ and for all $0 \leq j < k : (\Pi^{-1}_{-j}, \Gamma_R) \models_f rev(\psi_1)$ $\Leftrightarrow (\Pi^{-1}, \Gamma_R) \models_f rev(\psi_1 \,\mathbf{U}\, \psi_2)$
- **Y**: $(\Pi, \Gamma) \models_f \mathbf{Y}\psi \Leftrightarrow \Pi_{-1} \neq \texttt{und}$ and $(\Pi_{-1}, \Gamma) \models_f \psi \underset{Prop\ 2}{\Leftrightarrow} (\Pi^{-1}_1 \neq \texttt{und}$ and $(\Pi^{-1}_1, \Gamma_R) \models_f rev(\psi) \Leftrightarrow (\Pi^{-1}, \Gamma_R) \models_f \mathbf{X}\psi$.
- (**S**) $(\Pi, \Gamma) \models_f \psi_1 \,\mathbf{S}\, \psi_2 \Leftrightarrow$ there exists $k \geq 0$ s.t. $\Pi_{-k} \neq \texttt{und}$, $(\Pi_{-k}, \Gamma) \models_f \psi_2$ and for all $0 \leq j < k : (\Pi_{-j}, \Gamma) \models_f \psi_1 \underset{Prop\ 2}{\Leftrightarrow}$ there exists $k \geq 0$ s.t. $\Pi^{-1}_{+k} \neq \texttt{und}$, $(\Pi^{-1}_k, \Gamma_R) \models_f rev(\psi_2)$ and for all $0 \leq j < k : (\Pi^{-1}_j, \Gamma_R) \models_f rev(\psi_1)$ $\Leftrightarrow (\Pi^{-1}, \Gamma_R) \models_f rev(\psi_1 \,\mathbf{S}\, \psi_2)$.

B Proofs from Section 4

B.1 Proofs from Section 4.1

Definition 11. *Let Π be a trace assignment over M, we say that a trace assignment Π^{st} over M^{st} is a "stuttered" trace assignment of Π iff*

- *(In the mapping points the traces have the same assignments to V): for all $x_i \in Dom(\Pi) : \Pi^{st}(x_i) = (\sigma^{st}_i, map_l)$ where $(\sigma_i, l) = \Pi(x_i)$ and for all $0 \leq l < |\sigma| - 1$:for all $v \in V : \sigma^{st}_i, map_l \models\Leftrightarrow \sigma_i, l \models_f v$*
- *(the stuttered traces are aligned over observations (changes in Γ) up to the last observation of the trace with fewer observations): $\Pi^{st} \models \theta^{st}_{\Gamma}$*

where $map_l(\sigma)$ is the position of the l-th occurrence of state in σ (map_l when the trace is clear from the context).

Lemma 2 *Let Π be a trace assignment, ψ and φ_{st} be two q.f. HyperLTL properties. If φ_{st} occurs finitely amount of time in Π :*

$$\Pi \models \mathcal{R}_{\varphi_{st}}(\psi) \Leftrightarrow \Pi_{|\neg\varphi_{st}|} \models_f \psi.$$

where $\Pi_{|\phi|}$ is defined as follows: for all $x \in Dom(\Pi) : (\sigma_{|\phi|}, \#_{\phi}(\sigma, j)) = \Pi_{|\phi|}(x)$ where $(\sigma, j) = \Pi(x)$, $\#_{\phi}(\sigma, j)$ counts the occurrence of ϕ in σ up to position j and $\sigma_{|\phi|}$ is defined as the subtrace obtained removing the states violating ϕ.

Proof. The proof is an adaption of Lemma 1 of [10] by generalizing single trace to set of traces. Although the original Lemma assumes that φ_{st} occurs infinitely often, it is easy to extend the proof to the case in which only finite occurrences of φ_{st} appear. It is sufficient to extend the proof of $\mathbf{X}$ considering the ´´corner“ case in which we are at the last occurrence of φ_{st}; in that case we would just violate the rewritten formula $(\mathbf{X}(\neg\varphi_{st}\,\mathbf{U}\,(\varphi_{st}\wedge\mathcal{R}_{\varphi_{st}}(\psi))))$ adhering with the standard finite semantics.

Proposition 3 *Let $\langle x\rangle\alpha[x]$ be a formula and let Π and Π^{st} be a trace assignment and its stuttered counterpart.*
$(\Pi,\Gamma)\models_f\langle x\rangle\alpha[x]\Leftrightarrow\Pi^{st}\models\mathcal{R}_{\neg state[x]}(\alpha[x])$

Proof. Since $\alpha[x]$ is defined over trace variable x: $\Pi^{st}\models\mathcal{R}_{\neg s[x]}(\alpha[x])\Leftrightarrow\sigma^{st},i^{st}\models\alpha$ where $(\sigma^{st},i^{st})=\Pi^{st}(x)$. From Lemma 2 $\sigma^{st},i^{st}\models\alpha\Leftrightarrow\sigma,i\models_f\alpha$ where $(\sigma,i)=\Pi(x)$ and finally: $\sigma,i\models_f\alpha\Leftrightarrow\Pi\models_f\langle x\rangle\alpha[x]$.

Proposition 4 *Let ψ be a SC-HyperLTL q.f. formula and let Π and Π^{st} be a trace assignment and its stuttered counterpart.*
If $\Pi^{st}\models\theta_\Gamma[x_1]$ then $(\Pi,\Gamma)\models_f\psi\Leftrightarrow\Pi^{st}\models\mathcal{R}_{end^{st}\vee\neg\theta_\Gamma x_1}(\psi)$

Proof. To prove the Lemma, we apply an inductive approach over the size of the formula.

Base case: $(v[x])$: $(\Pi,\Gamma)\models_f v[x]\Leftrightarrow v\in\sigma(l)$ with $(\sigma,l)=\Pi(x)$. By assumption $(\sigma^{st},map_l)=\Pi^{st}(x)$ and $\sigma^{st}(map_l)_{|V|}=\sigma(l)$; thus, $\cdots\Leftrightarrow\Pi^{st}\models v[x]$

Inductive case:

- $(\vee,\neg)$ Trivially follows from induction and operator semantics
- $(\mathbf{X})$: We can split the proof in two cases: $succ_\Gamma(\Pi)=\texttt{und}$ or $succ_\Gamma(\Pi)\neq\texttt{und}$. If the successor is undefined, then there is one trace variable x s.t. $(\sigma,|\sigma|-1)=\Pi(x)$. Thus, by assumption $\Pi^{st}\models\mathbf{X}halt[x]$. Since in M^{st} *halt* is only present in sink states that loops forever, then $\mathcal{R}_{end^{st}\vee\neg\theta_\Gamma x_1}(\mathbf{X}\psi)$ is violated. If $succ_\Gamma(\Pi)\neq\texttt{und}$, then by definition $succ_\Gamma(\Pi)$ points to the first occurrence after the current one of each trace in which θ_Γ is true. Due to the assumptions on alignment of definition 11 then such positions are exactly in the first point s.t. $\theta_\Gamma[x_1]$ is true i.e. $\cdots\Leftrightarrow$ exists $k\geq0$ s.t. $\Pi^{st}+k\models\neg end^{st}\wedge\theta_\Gamma[x_1]$, $\Pi^{st}+k\models\mathcal{R}_{\dots}(\psi_2)$ and for all $0\leq j<k:\Pi^{st}+j\models end^{st}\vee\neg\theta_\Gamma[x_1]\Leftrightarrow\mathcal{R}_{end^{st}\vee\neg\theta_\Gamma[x_1]}(\mathbf{X}\psi)$
- $(\mathbf{U})$: $(\Pi,\Gamma)\models_f\psi_1\mathbf{U}\psi_2\Leftrightarrow$ exists $k\geq0$ s.t. $succ^k_\Gamma(\Pi)\neq\texttt{und}$, $(succ^k_\Gamma(\Pi),\Gamma)\models_f\psi_2$ and for all $0\leq j<k:(succ^j_{(\Gamma)}(\Pi),\Gamma)\models_f\psi_1$. Since *succ* is defined iff $\Pi^{st}\models\neg end^{st}$ and as for $\mathbf{X}$, we know that Γ-successors in Π^{st} must satisfy $\theta_\Gamma[x_1]$. Thus, $\cdots\Leftrightarrow$ exists $k'\geq0$ s.t. $\Pi^{st}\models\mathcal{R}_{end^{st}\vee\neg\theta_\Gamma[x_1]}(\psi_2)\wedge\neg end^{st}\wedge\theta_\Gamma[x_1]$ and for all $0\leq j'<k':\Pi^{st}+j'\models\mathcal{R}_{end^{st}\vee\neg\theta_\Gamma[x]}(\psi_1)\vee end^{st}\vee\theta_\Gamma[x_1]\Leftrightarrow\mathcal{R}_{\dots}(\psi_1\,\mathbf{U}\,\psi_2)$
- $(\langle x\rangle\alpha[x])$: Follows from proposition 3.
- (past operators): simplified version of the future.

Proof of Theorem 3

Proof.

$$M \models_f \mathcal{Q}x_n \dots \mathcal{Q}x_1.\Gamma.\psi \Leftrightarrow M^{st} \models \mathcal{Q}x_n \dots \mathcal{Q}x_1.\theta_\Gamma^{st} \odot_{\mathcal{Q}} \mathcal{R}_{end^{st} \vee \neg\theta_\Gamma[x_1]}(\psi)$$

– ($\Rightarrow$) Suppose that $M \models_f \mathcal{Q}x_n \dots \mathcal{Q}x_1.\Gamma.\psi$ and $M^{st} \not\models \mathcal{Q}x_n \dots \mathcal{Q}x_1.\theta_\Gamma^{st} \odot_{\mathcal{Q}} \mathcal{R}_{end^{st} \vee \neg\theta_\Gamma[x_1]}(\psi)$.
 We consider case $\mathcal{Q} = \forall$: Then, there must be a trace Π^{st} s.t. $\Pi^{st} \models \theta_\Gamma^{st} \wedge \neg\mathcal{R}_{end^{st} \vee \neg\theta_\Gamma[x_1]}(\psi)$. Since Π^{st} is a trace of M^{st}, there exists a trace assignment that is defined as $\Pi^{st}_{|state|}$ s.t. Π^{st} is the "stuttered" trace assignment of $\Pi^{st}_{|state|}$. Then, by Proposition 4 $\Pi := \Pi^{st}_{|state|} \models_f \neg\psi$. Since Π is a trace assignment generated removing stuttering and sink transition, we obtain that it is constructed by traces of M. However, by hypothesis $M \models_f \forall x_n. \dots \forall x_1.\Gamma.\psi$ while $\Pi \models_f \neg\psi$ (contradiction). The converse (exists) is equivalent.
– ($\Leftarrow$): The opposite direction is specular.

B.2 Proofs from Section 4.2

In this section, we introduce the following notations that we are going to use in the proofs.

Let M be a STS, $\alpha := \mathcal{Q}_n x_n \dots \mathcal{Q}_1 x_1.\Gamma.\psi$ be a SC-HyperLTL formula where Γ is a set of LTL formulas over $\overline{V} \subseteq V$.

Let $\tilde{\sigma}_1, \dots \tilde{\sigma}_n \in \mathsf{Traces}^{fin}(M)$ be n traces of M. Let $\tilde{\Pi}$ be the trace assignment s.t. for all $x_j \in$ VAR: $\tilde{\Pi}(x_j) = (\tilde{\sigma}_j, 0)$. Finally, we denote $\tilde{\Pi}_i := succ^i_\Gamma(\tilde{\Pi})$.

Proof of Prop. 1

Definition 12. *Let φ be an LTLf property, V be the set of variables occurring in φ and $v_\varphi \notin V$ be a fresh Boolean variable. We define M_φ as the STS s.t. (i) $M \models_f \varphi \Leftrightarrow M_\varphi \models_f v_\varphi$, (ii) the set of traces of M_φ restricted to the variables V is exactly the set of traces of M.*

Proof. We prove the proposition considering the singleton context case: we consider $\tilde{\Pi}$ as the trace assignment of M and Π as the corresponding assignment over $M \times \bigtimes_{\langle x_i \rangle \beta[x_i] \in Sub(\varphi)} M_\beta$. $\tilde{\Pi} \models_f \langle x_i \rangle \beta \Leftrightarrow (\tilde{\sigma}_i, j) \models_f \beta$ with $(\tilde{\sigma}_i, j) = \tilde{\Pi}(x_i)$. By definition then $\dots \Leftrightarrow (\sigma_i, j) \models_f v_\beta$ where $(\sigma_i, j) = \Pi(x_i)$, from which we obtain $\dots \Leftrightarrow \Pi \models_f v_\beta$ proving the claim.

Proving Theorem 4

Proposition 5 *For all $\tilde{\sigma} \in \mathsf{Traces}^{fin}(M)$, $\sigma \in \mathsf{Traces}^{fin}(M^k_{\mathcal{B}})$ s.t. $\sigma(V) = \tilde{\sigma}$ (recall that traces of M are preserved in $M^k_{\mathcal{B}}$), for all $v \in \overline{V}$, for all $0 \leq i < k$:*

$$\text{If } \sigma, |\sigma| - 1 \models_f pos > i, \text{then } \begin{pmatrix} \sigma, |\sigma| - 1 \models_f v_i \Leftrightarrow \\ succ^i_\Gamma(\tilde{\sigma}, 0) \models_f v \end{pmatrix}$$

Moreover, $\sigma, |\sigma| - 1 \models_f pos \leq i : succ^i_\Gamma(\tilde{\sigma}, 0) = \mathtt{und}$.

Proof. We prove the proposition by considering the following claims that combined entail it. Let

i For all $0 \leq j < |\sigma| : \sigma, j \models_f pos = j \wedge obs_\Gamma \Leftrightarrow (\tilde{\sigma}, j) = succ^i_\Gamma(\tilde{\sigma}, 0)$
ii Let $(\tilde{\sigma}, j) = succ^i_\Gamma(\tilde{\sigma}, 0)$:
  a $(\sigma, j) \models_f v \Leftrightarrow (\sigma, j+1) \models_f v_i$
  b For all $j' > j : (\sigma, j) \models_f v_i \Leftrightarrow (\sigma, j') \models_f v_i$

The first claim can be easily claimed by induction on i:

$(i = 0)$: $succ^0_\Gamma(\tilde{\sigma}, 0) = (\tilde{\sigma}, 0)$, Since $pos = 0$ at the initial state (due to $P^k_{\theta_\Gamma, \overline{V}}$) and obs_Γ is true if $pos = 0$ then one direction is proved. The other direction follows because pos is increased when obs_Γ holds and it never decrease; that means that pos is 0 initially, then it become 1 and then it remain weakly monotonic.

$(i - 1 \Rightarrow i)$: We know that $succ^i_\Gamma(\tilde{\sigma}, 0) := succ_\Gamma(succ^{i-1}_\Gamma(\tilde{\sigma}, 0))$. By induction we know that either $succ^{i-1}_\Gamma(\tilde{\sigma}, 0) = \texttt{und}$ or there is a unique point j s.t. $(\sigma, j) = succ^{i-1}_\Gamma(\tilde{\sigma}, 0)$ and $\sigma, j \models_f obs_\Gamma \wedge pos = i - 1$.
If $succ^{i-1}_\Gamma(\tilde{\sigma}, 0) = \texttt{und}$, then $succ^i_\Gamma(\tilde{\sigma}, 0) = \texttt{und}$ as well; since pos changes only by increasing by one exactly in the points in which obs_Γ is true, then if there is no point s.t. $obs_\Gamma \wedge pos = i-1$ is true, there is not point s.t. $obs_\Gamma \wedge pos = i$ as well.
If $succ^{i-1}_\Gamma(\tilde{\sigma}, 0) = (\sigma, j)$, then $succ_\Gamma(\tilde{\sigma}, j) = succ^i_\Gamma(\tilde{\sigma}, 0)$. Since obs_Γ captures the semantics of observation points (initial state, tail states or points in which previously $\theta \in \Gamma$ changed) then either $succ^i_\Gamma(\tilde{\sigma}, 0) = \texttt{und}$ or $succ^i_\Gamma(\tilde{\sigma}, 0) = (\sigma, j')$ s.t. $j' > j$ and for all $j < j'' < j' : \sigma, j'' \not\models_f obs_\Gamma$. Since pos changes only if obs_Γ is true, then either $pos = i \wedge obs_\Gamma$ is never true in the trace or it will be true exactly at point j' with $(\sigma, j') = succ^i_\Gamma(\sigma, 0)$.

Claim ii instead follows by the definition of $P^k_{\theta_\Gamma, \overline{V}}$. It constraints v_i to take the value of v at the i-th observation and then it make v_i stutters forever (because $pos = i \wedge obs_\Gamma$ is true at most once).

Given the two claims, the first part of the proposition follows from claim i and part 2 of claim ii. The second part of the proposition follows from claim i.

Lemma 3 *Let $\sigma_1, \ldots, \sigma_n$ be n traces of $M^k_{\mathcal{B}}$ and $\tilde{\sigma}_1, \ldots, \tilde{\sigma_n}$ the corresponding trace of M (i.e. traces s.t. For all $0 \leq i < |\tilde{\sigma}|$: $\sigma(i)(V) = \tilde{\sigma}(i)(V)$ and $\sigma, |\tilde{\sigma}| \models_f end$), let Π be trace assignment over VAR s.t. for all $x_i \in VAR : \Pi(x_i) := (\sigma_i, 0)$, let $\tilde{\Pi}$ be the trace assignment over VAR s.t. for all $x_i \in VAR : \Pi(x_i) := (\tilde{\sigma}, 0)$ and let ψ be a q.f. SC-HyperLTL formula without singleton contexts.*

If there is a $\tilde{\sigma}_i$ s.t. θ_Γ occurs at most k times:

$$(\tilde{\Pi}, \Gamma) \models_f \psi \Leftrightarrow \Pi \models_f \mathbf{G}(\mathbf{N}\bot \rightarrow {}^\rho[\![\psi]\!]^k_0)$$

Proof. We prove the Lemma using the following statement: For all i: $\tilde{\Pi}_i \neq \texttt{und} \Rightarrow \tilde{\Pi}_i \models_f \psi \Leftrightarrow \Pi^e \models_f {}^\rho[\![\psi]\!]^k_i$ where Π^e points each σ to position $|\sigma| - 1$. The statement can be proved by induction on the structure of the formula.

Base case: $\tilde{\Pi}_i \models_f v[x] \Leftrightarrow succ^i_{\Gamma}(\tilde{\sigma}, 0) \models_f v$. By Proposition 5, if $\sigma, |\sigma| - 1 \models_f pos > i$, then $\sigma, |\sigma| - 1 \models_f v_i \Leftrightarrow succ^i_{\Gamma}(\tilde{\sigma}, 0) \models_f v$. Therefore, we deduce that $\sigma, |\sigma - 1| \models_f v_i \Leftrightarrow succ^i_{\Gamma}(\tilde{\sigma}, 0) \models_f v$.

We can ignore the base case with $i = k$ since by assumption $\Pi_{k+1} = \texttt{und}$, thus ${}^{\rho}[\![\psi]\!]^k_k$ is never reached (to be reached *pos* must be $= k + 1$ in each trace).

Inductive case:

- $(\vee, \wedge)$: Trivial induction.
- $\mathbf{X}\psi$: $(\tilde{\Pi}_i, \Gamma) \models_f \mathbf{X}\psi \Leftrightarrow \tilde{\Pi}_{i+1} \neq \texttt{und}$ and $(\tilde{\Pi}_{i+1}, \Gamma) \models_f \psi$.
  By Proposition 5 and due to the semantics of the successor, we know that $\tilde{\Pi}_{i+1} \neq \texttt{und} \Leftrightarrow \Pi^e \models_f \neg end^{i+1}_{\Gamma}$. Moreover, by induction if $\tilde{\Pi}_{i+1} \neq \texttt{und}$: $(\tilde{\Pi}_{i+1}, \Gamma) \models_f \psi \Leftrightarrow \Pi^e \models_f {}^{\rho}[\![\psi]\!]^k_{i+1}$. Therefore, we can conclude that $(\tilde{\Pi}_i, \Gamma) \models_f \mathbf{X}\psi \Leftrightarrow \Pi^e \models_f \neg end^{i+1}_{\Gamma} \wedge {}^{\rho}[\![\psi]\!]^k_{i+1} \Leftrightarrow \Pi \models_f {}^{\rho}[\![\mathbf{X}\psi]\!]^k_i$.
- $\mathbf{N}_{\Gamma}\psi$: The proof is identical to the one of $\mathbf{X}$ with the minor difference that $\Pi_{i+1} \neq \texttt{und}$ and $\neg end^{i+1}_{\Gamma}$ appear with an implication instead of a conjunction.
- $\psi_1 \mathbf{U} \psi_2$: $(\tilde{\Pi}_i, \Gamma) \models_f \psi_1 \mathbf{U} \psi_2 \Leftrightarrow$ exists $l \geq i$ s.t. $\tilde{\Pi}_l \neq \texttt{und}$, $(\tilde{\Pi}_l, \Gamma) \models_f \psi_2$ and for all $i \leq j < l$: $(\tilde{\Pi}_j, \Gamma) \models_f \psi_1$. Combining Proposition 5, due to the successor semantics (same as $\mathbf{X}$) and using induction, we obtain $\dots \Leftrightarrow \Pi^e \models_f \bigvee_{i \leq l \leq k}(\neg end^i_{\Gamma} \wedge {}^{\rho}[\![\psi_2]\!]^k_l \wedge \bigwedge_{i \leq j < l}({}^{\rho}[\![\psi_1]\!]^k_j))$. We observe that the previous expression is equivalent to the unrolling: ${}^{\rho}[\![\psi_2]\!]^k_i \vee {}^{\rho}[\![\psi_1]\!]^k_i \wedge \neg end^{i+1}_{\Gamma} \wedge {}^{\rho}[\![\psi_1 \mathbf{U} \psi_2]\!]^k_{i+1} (\Leftrightarrow {}^{\rho}[\![\psi_1 \mathbf{U} \psi_2]\!]^k_i)$. Thus, we can conclude that $(\tilde{\Pi}_i, \Gamma) \models_f \psi_1 \mathbf{U} \psi_2 \Leftrightarrow \Pi^e \models_f {}^{\rho}[\![\psi_1 \mathbf{U}_{\Gamma} \psi_2]\!]^k_i$.
- $\psi_1 \mathbf{R} \psi_2$: The proof for release follows a similar procedure of until.
- $\mathbf{Y}\psi$: We split the two cases:
  1. $i = 0$: By definition $\tilde{\Pi}_i \not\models_f \mathbf{Y}\psi$. By ${}^{\rho}[\![\mathbf{Y}_{\Gamma}\psi]\!]^k_0$ definition, $\Pi^e \not\models_f {}^{\rho}[\![\mathbf{Y}_{\Gamma}\psi]\!]^k_0$.
  2. $0 < i < k$: $\tilde{\Pi}_i \models_f \mathbf{Y}_{\Gamma}\psi \Leftrightarrow \tilde{\Pi}_{i-1} \neq \texttt{und}$ and $\tilde{\Pi}_{i-1} \models_f \psi$. Since $i > 0$, $\tilde{\Pi}_{i-1} \neq \texttt{und}$. Therefore, we obtain $\tilde{\Pi}_i \models_f \mathbf{Y}\psi \Leftrightarrow \tilde{\Pi}_{i-1} \models_f \psi$. By induction we obtain $\dots \Leftrightarrow \Pi^e \models_f {}^{\rho}[\![\psi]\!]^k_{i-1}$ proving the equivalence.
- $\mathbf{Z}\psi$: Proof almost identical to yesterday (case 1 is true instead of false).
- $\psi_1 \mathbf{S} \psi_2$: $(\tilde{\Pi}_i, \Gamma) \models_f \psi_1 \mathbf{S} \psi_2 \Leftrightarrow$ exists $l \leq i$ s.t. $(\tilde{\Pi}_l, \Gamma) \neq \texttt{und}$, $\tilde{\Pi}_l \models_f \psi_2$ and for all $l < j \leq i$: $(\tilde{\Pi}_j, \Gamma) \models_f \psi_1 \Leftrightarrow \Pi^e \models_f \bigvee_{0 \leq l \leq k}({}^{\rho}[\![\psi_2]\!]^k_l \vee \bigwedge_{l < j \leq i} {}^{\rho}[\![\psi_1]\!]^k_j)$. We observe that the previous expression is equivalent to the unrolling: ${}^{\rho}[\![\psi_2]\!]^k_i \vee {}^{\rho}[\![\psi_1]\!]^k_i \wedge {}^{\rho}[\![\mathbf{Y}\top]\!]^k_{i-1} \wedge {}^{\rho}[\![\psi_1 \mathbf{S} \psi_2]\!]^k_{i-1}$.
- $\psi_1 \mathbf{T} \psi_2$: Proof almost identical to since

Finally, we need to prove that $\Pi^e \models_f {}^{\rho}[\![\psi]\!]^k_i \Leftrightarrow \Pi \models_f \mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\rho}[\![\psi]\!]^k_i)$.

We recall that $\mathbf{N}\bot$ is satisfied by a trace assignment at the end of the shortest trace. Since all traces of $M^k_{\mathcal{B}}$ terminates in a state s.t. *end* is true and when *end* is true the variables remain unchanged $((end' \rightarrow V' = V)$ and the fact that obs_{Γ} requires $\neg end$.

Proof of the first part of Theorem 4

Proof. Since $M^k_{\mathcal{B}} \models_f \mathbf{G}(pos \leq k)$ each trace of $M^k_{\mathcal{B}}$ has at most k observations.

We can prove the first part of the theorem by induction with quantifiers and Γ. The statement is the following: Let $\tilde{\Pi}$ be the partial trace assignment of $x_n, \ldots, x_{i+1}$ over M and Π be the partial trace assignment of $x_n, \ldots, x_i + 1$ over $M^k_{\mathcal{B}}$ s.t. each trace of $\tilde{\Pi}$ correspond with the extended trace with end of $M^k_{\mathcal{B}}$. Then, $\Pi \models_f \mathcal{Q}_i x_i \ldots \mathcal{Q}_1 x_1.\psi^i \Leftrightarrow \Pi' \models_f \mathcal{Q}_i x_i \ldots \mathcal{Q}_1 x_1.\psi$.

Base case(Γ): $\tilde{\Pi} \models_{f_T} \Gamma.\varphi \Leftrightarrow (\tilde{\Pi}, \Gamma) \models_f \varphi$. By Lemma 3 we obtain $\tilde{\Pi} \models_{f_T} \Gamma.\varphi \Leftrightarrow \Pi \models_f \mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\rho}[\![\varphi]\!]^k_0$.

Inductive case: $\tilde{\Pi} \models_{f_T} \forall x_i.\alpha \Leftrightarrow$ for all $\tilde{\sigma}_i \in \mathsf{Traces}^{fin}(M) \tilde{\Pi}[x_i \mapsto \tilde{\sigma}_i] \models_{f_T} \alpha$. By induction we obtain $\Pi[x_i \mapsto \sigma_i \models_{f_T} \mathcal{Q}_{-1} \ldots .\mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\rho}[\![\varphi]\!]^k_0)$. Since each trace $\tilde{\sigma} \in \mathsf{Traces}^{fin}(M)$ as a corresponding trace $\sigma \in \mathsf{Traces}^{fin}(M^k_{\mathcal{B}})$ and vice-versa, we obtain: for all $\tilde{\sigma}_i \in \mathsf{Traces}^{fin}(M) \tilde{\Pi}[x_i \mapsto \tilde{\sigma}_i] \models_{f_T} x_{i-1} \ldots .\Gamma.\varphi \Leftrightarrow$ for all $\sigma_i \in \mathsf{Traces}^{fin}(M^k_{\mathcal{B}})$ $\Pi[x_i \mapsto \sigma_i] \models_{f_T} x_{i-1} \ldots .\mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\rho}[\![\varphi]\!]^k_0) \Leftrightarrow \Pi \models_f \forall x_i.\mathcal{Q}_{i-1} \ldots .\mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\rho}[\![\varphi]\!]^k_0)$. The proof for the existential case is identical.

Proof of the second part of Theorem 4

Proof. Let $\alpha := \mathcal{Q}_n x_n \ldots \mathcal{Q}_1 x_1.\Gamma.\varphi$ and let ${}_{\rho}\alpha^k_B := \mathcal{Q}_n x_n \ldots \mathcal{Q}_1 x_1.\mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\rho}[\![\varphi]\!]^k_0)$.

Suppose by w.o.c. that $M^k_{\mathcal{B}} \models_f {}^{-}\alpha^k_B$ and $M \not\models_f \alpha$.

We consider for simplicity trace assignments Π (for $M^k_{\mathcal{B}}$) and $\tilde{\Pi}$ (for M).

Since $(\tilde{\Pi}, \Gamma)$ violates φ, then there must be more than k' observations in $\tilde{\Pi}$ and Π s.t. $k' > k$ (note that k' is finite but unbounded).

Let us now considered the trace assignment Π' obtained from $\tilde{\Pi}$ to be part of $M^{k'}_{\mathcal{B}}$ (pos is determined deterministically while v_i are non-deterministically chosen until $pos = i \wedge obs_{\Gamma}$ is true).

We derive that $\Pi' \not\models_f \mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\bot}[\![\varphi]\!]^{k'}_0)$. By the unrolling structure of ${}^{\rho}[\![\varphi]\!]^k_0$, we see that $\Pi' \models_f \mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\bot}[\![\varphi]\!]^k_0) \Rightarrow \Pi' \models_f \mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\bot}[\![\varphi]\!]^{k'}_0)$. This is true because the unrolling at k is false with ${}^{\bot}[\![\varphi]\!]^k_0$ and negation occur only in the leafs (φ is in nnf).

Finally, $\Pi' \not\models_f \mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\bot}[\![\varphi]\!]^k_0)$ implies that $\Pi \not\models_f \mathbf{G}(\mathbf{N}\bot \rightarrow {}^{\bot}[\![\varphi]\!]^k_0)$ because both traces have the same assignments of V, the same assignments of prophecies variables up to k, the same assignment to pos (until it reaches $k+1$), and the property does not depends on $v_{i+1}, \ldots v_k$ (for each $v \in \overline{V}$) or pos when it is assigned to values greater than $k+1$ (we just check it to be $> k$). Therefore, we encounter a contradiction.

The proof of the other direction is very similar.