EPiC Series in Computing

Volume 110, 2026, Pages 1–10

Proceedings of 13th Int. Workshop on Applied Verification for Continuous and Hybrid Systems


# SpaceEx Hybrid Models with LTL Properties

Ludovico Battista, Stefano Tonetta, and Gianni Zampedri

Fondazione Bruno Kessler, Trento 38123, Italy

**Abstract**

The automated verification of hybrid systems has traditionally focused on safety and reachability properties. However, verifying liveness and general Linear Temporal Logic (LTL) properties is essential in many applications. This paper proposes a new benchmark suite for the ARCH competition, utilizing an LTL-enhanced extension of the standard SpaceEx model format. Our suite consists of several hybrid automata featuring linear and non-linear dynamics, ranging from 2D scalable models to 10D vehicle drivetrain systems. We provide a detailed formalization of a car overtaking maneuver to demonstrate how temporal specifications are encoded directly within the model files. To establish a baseline for the competition, we present experimental results obtained using the VeriLHyS tool. These benchmarks are intended to serve as a foundation for cross-tool comparisons and to foster the development of LTL support in other tools.

## 1 Introduction

Verification of hybrid systems remains a central challenge for safety-critical applications, where continuous physical dynamics must interact reliably with discrete control logic. While several tools and techniques have been proposed over the years (see for instance [FGD+11, CÁS13, Alt15] [BD17, SÁMK17]), the evaluation and comparison of these approaches strongly depends on the availability of benchmarks. If safety and reachability properties have received significant attention, the automated verification of liveness has been tackled only for specific types of evolution and properties.

To address this gap, this paper introduces a new benchmark suite defined in an LTL-enhanced extension of the SpaceEx [FGD+11] format, supported by the VeriLHyS tool [BTZ]. The benchmarks are publicly available at https://es-static.fbk.eu/tools/verilhys/[1] and are meant to serve both as regressions tests and as a basis for cross-tool comparisons.

This paper is structured as followed. Section 2 describes the LTL-enhanced SpaceEx format used by VeriLHyS, section 3 provides an overview of a set of benchmark models to be used for verification of hybrid systems, focusing in detail on the *Linear Overtake* example; lastly, section 4 shows some experimental results obtained with VeriLHyS to be served as a baseline.

[1]The benchmarks will be made available on the ARCH website if accepted

# 2 LTL-Enhanced SpaceEx Format

To specify the benchmakrs we adopt an input syntax based on the SpaceEx [FGD+11] model format, widely used for representing hybrid automata with linear or affine dynamics. To support properties expressed in Linear Temporal Logic (LTL), the format is extended with additional constructs that allow users to declare LTL formulae to be checked.

Formally, the input grammar, expressed in Compact RelaxNG format, is enriched with the following additions:

- `initial_location`. A tag used to specify the initial location of the system. The (integer) value must correspond to a location defined within the model's location list.
- `initial_region`. A tag used to define the initial region of the system's state space. Classical arithmetic operations are allowed inside it.
- `invariant`, `reach` and `ltl-property`. These elements allow the specification of invariants ($Gp$), reachability ($Fp$) and general LTL formulae, respectively, which are used during system analysis. The following LTL operators are supported: `X`, `G`, `F`, `U`, `R`

Another difference with respect to the native SpaceEx language, is given by the fact that this format enforces syntactic and semantic validation of expressions embedded with the standard tags `flow`, `guard` and `reset-relation`.

A complete description of the grammar defining the format is publicly available on VeriLHyS website at https://es-static.fbk.eu/tools/verilhys/documentation.html.

The semantics of the represented hybrid system in terms of hybrid traces and the interpretation of LTL formulas over such traces is recalled in the background section of [BT25].

# 3 Benchmarks

## 3.1 Benchmarks Suite Overview

The benchmark suite consists of several hybrid automata encoded in the LTL-enhanced SpaceEx format, which are available on the VeriLHyS website.

The following benchmarks are included in the suite:

**Linear overtake** A 4-dimensional linear hybrid system modeling overtaking behavior of a vehicle.

**Linear overtake relaxed** A variant of the previous 4-dimensional linear hybrid system modeling overtaking behavior of a vehicle, in which guards are rewritten using inequalities.

**Multiple choice** A scalable 2D benchmark with discrete jumps.

**Navigation** A 4D system (space and velocity variables) with linear dynamics and 25 locations.

**Non-linear circular** A 2D system with non-linear dynamics converging to the unit circle.

**Multilayer control linear** A scalable 3D system with multiple locations converging with linear dynamics to different points in the space.

**Multilayer control nonlinear** A scalable 3D system with multiple locations converting with nonlinear dynamics to different points in the space.

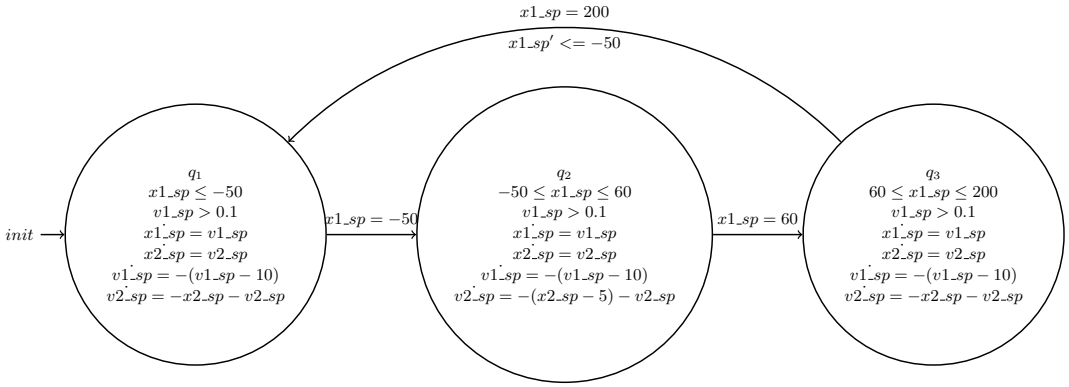


Figure 1: Linear Overtake Model

**Bball** 2D linear systems representing a bouncing ball.

**Circle** A 2D linear system with two locations representing circular motion.

**Drivetrain** A 10D linear system modeling a vehicle's drivetrain, available in 2 different versions.

**Rendezvous** A 4D nonlinear system modeling a spacecraft rendezvous maneuver. Available in 6 different versions.

Part of the benchmark suite (i.e. *Bball*, *Circle*, *Drivetrain* and *Rendezvous*) is derived from the existing hybrid-automaton examples encoded in SpaceEx format distributed with CORA[2][Alt15]; these models have been converted into the LTL-enhanced SpaceEx format by retaining their original dynamics and structure while enriching them with atomic propositions and LTL properties.

Among the available benchmarks, we provide a detailed description of the *linear_overtake* model. We use this example to illustrate how hybrid dynamics and temporal specifications are encoded in the LTL-enhanced SpaceEx format. By presenting it in full, we aim to clarify how model elements, atomic propositions, and LTL properties are represented.

The remaining benchmarks follow the same encoding principles and are therefore summarized in a compact form, while their complete model files are included in the supplementary material.

## 3.2 Linear Overtake

This benchmark describes a 4-dimensional linear hybrid automaton modeling a simplified vehicle-overtaking maneuver on a straight road. The model captures the longitudinal and lateral motion of an ego vehicle as it approaches, overtakes, and returns to lane after passing a slower vehicle occupying a fixed region $B$ on the road.

The maneuver is divided into three phases:

- q1 = Standard: ego vehicle approaches the obstacle vehicle.

[2]The original models are available at: https://github.com/TUMcps/CORA/tree/master/models/SpaceEx

- q2 = Overtaking: ego vehicle changes lateral position and passes the obstacle.
- q3 = Standard_after: ego vehicle returns to its lane after completing the overtake.

This hybrid system uses piecewise affine continuous dynamics, and is well-suited to specify different LTL (both safety and liveness) properties.

### 3.2.1 System description

The system exposes four real-valued continuous variables:

- x1_sp: longitudinal position of the ego vehicle relative to the obstacle vehicle
- x2_sp: lateral position of the ego vehicle
- v1_sp: longitudinal velocity of the ego vehicle.
- v2_sp: lateral velocity of the ego vehicle.

Listing 1: VeriLHyS format - Declaration of variables

```
<param name="x1_sp" type="real" local="false"
       dynamics="any" d1="1" d2="1" />
<param name="x2_sp" type="real" local="false"
       dynamics="any" d1="1" d2="1" />
<param name="v1_sp" type="real" local="false"
       dynamics="any" d1="1" d2="1" />
<param name="v2_sp" type="real" local="false"
       dynamics="any" d1="1" d2="1" />
```

All dynamics in the model are first-order linear ODEs.
The hybrid automaton has three locations:

1. `Standard`: Represents normal following behavior before the overtake starts. Its invariant is defined as
$$x1\_sp \leq 50 \wedge v1\_sp > 0.1$$
2. `Overtake`: Ego vehicle moves laterally to pass the obstacle and then continues forward while displaced laterally. Its invariant is defined as
$$-50 \leq x1\_sp \leq 60 \wedge v1\_sp > 0.1$$
3. `Standard_after`: Ego vehicle has passed the obstacle and returns to a normal lane position. Its invariant is defined as
$$60 \leq x1\_sp \leq 200 \wedge v1\_sp > 0.1$$

Each location defines linear dynamics of the form

$$x1\_sp' = v1\_sp, x2\_sp' = v2\_sp, v1\_sp' = 10 - v1\_sp, v2\_sp' = -v2\_sp - x2\_sp + \mu$$

where the constant input $\mu$ depends on the phase:

- Standard: $\mu = 0$
- Overtaking: $\mu = 5$
- Standard_after: $\mu = 0$

These dynamics cause $v1\_sp$ to converge towards the desired speed 10, and $(x2\_sp, v2\_sp)$ to converge to either the center of the lane ($\mu = 0$) or a displaced lateral position ($\mu = 5$).

Listing 2: VeriLHyS format - Encoding `Standard` location

```
<location id="0" name="Standard">
    <invariant>
        (x1_sp &lt;= -50) &amp; (v1_sp &gt; 1/10)
        </invariant>
    <flow>
        x1_sp' == v1_sp &amp; x2_sp' == v2_sp &amp;
        v1_sp' == 10 - v1_sp &amp; v2_sp' == -v2_sp - x2_sp
    </flow>
</location>
```

The automaton contains three deterministic transitions:

1. `Standard` to `Overtaking`. Triggered when the ego vehicle is behind the obstacle and the distance is exactly 50 ($x1\_sp = 50$)
2. `Overtaking` to `Standard_after`. Triggered when the ego vehicle has moved ahead of the obstacle and the distance is exactly 60 ($x1\_sp = 60$).
3. `Standard_after` to `Standard`. Triggered when the overtaking has been completed ($x1\_sp = 200$) with reset placing the vehicle back to a starting longitudinal interval $x1\_sp' \leq -50$.

All transitions leave velocities and lateral position unchanged.

Listing 3: VeriLHyS format - Encoding of `Standard` to `Overtaking` transition

```
<transition source="0" target="1">
    <guard>
        x1_sp == -50
    </guard>
    <assignment>
        x1_sp := x1_sp &amp; x2_sp := x2_sp &amp;
        v1_sp := v1_sp &amp; v2_sp := v2_sp
    </assignment>
</transition>
```

The initial region enforces:

- ego vehicle is far behind obstacle: $x1\_sp \leq -100$,
- lateral displacement close to lane center: $\|x2\_sp\| \leq 0.1$,
- longitudinal speed near desired cruising speed: $9.9 \leq v1\_sp \leq 10.1$
- lateral velocity small: $\|v2\_sp\| < 0.1$

Listing 4: VeriLHyS format - Definition of the initial region

```
<initial_region>
    (v1_sp &gt;= 99/10) &amp; (x2_sp &gt;= -1/10) &amp;
    (v1_sp &lt;= 101/10) &amp; (x1_sp &lt;= -100) &amp;
    (x2_sp &lt;= 1/10) &amp; (v2_sp &gt; -1/10) &amp; (v2_sp &lt; 1/10)
</initial_region>
```

This models the ego vehicle in steady-state cruising before the overtaking event.

Finally, the model specifies the temporal properties to be verified by VeriLHyS. In *linear_overtake*, we consider the following three properties:

1. "From every reachable state, the system must eventually reach the location `Standard` again". In other words, this is used to check that the overtaking maneuver cannot get stuck in `Overtaking` or `Standard_after`. This is a liveness property ensuring the overtaking behavior completes successfully.
2. "Whenever the system is in `Standard`, it must avoid a set of unsafe conditions until the location `Standard_after` is reached." Unsafe conditions include lateral deviation beyond allowed bounds, and entering the region in front of the obstacle with insufficient lateral displacement.
3. "Globally, the ego vehicle must never enter any of several unsafe geometric regions relative to the obstacle". These regions encode excessive lateral deviation and being too close longitudinally to the obstacle while still too close to the lane center during overtaking.

Listing 5: VeriLHyS format - Encoding of LTL properties

```
<ltl-property>
    G ( F ( Standard ) )
</ltl-property>
```

### 3.3 Additional benchmarks

In addition to the *linear_overtake* benchmark detailed in the previous section, the suite contains several further models. Because these benchmarks follow similar modeling patterns, their key structural features are summarized in table 1, while full encodings are provided in the supplementary materials on the VeriLHyS website.

## 4 Results with VeriLHys

In this section, we report the verification results obtained by running VeriLHyS on all benchmarks described above.

Table 2 lists, for each benchmark, LTL properties we proved; our approach results to be effective, as it can be inferred from the variety of properties verified. The algorithm has been tested with four configuration: with/without images through jumps and with/without stabilizing constraints[3]; all the shown properties were proved in at least one configuration.

[3]Supplementary details at https://es-static.fbk.eu/people/lbattista/pages/verilhys_tool.html

| **Benchmark** | **Real-value variables nbr.** | **Locations nbr.** | **Flow** |
|---|---|---|---|
| Linear overtake | 4 | 3 | Lin |
| Linear overtake relaxed | 4 | 3 | Lin |
| Multiple choice | 2 | 2 + 2n | Lin |
| Navigation | 4 | 25 | Lin |
| Non-linear circular | 2 | 1 | Nonlin |
| Multilayer control lin | 3 | n | Lin |
| Multilayer control nonlin | 3 | n | Nonlin |
| Bball | 2 | 1 | Lin |
| Circle | 2 | 2 | Lin |
| Drivetrain | 10 | 4 | Lin |
| Rendezvous | 4 | 1 to 3 | Nonlin |

Table 1: Benchmarks Suite Overview

| **Benchmark** | **Properties** |
|---|---|
| Linear overtake | $G(q_1 \rightarrow ((\neg(B))Uq_3))$, $G(F(q_1)), G(\neg B)$ |
| Linear overtake relaxed | $G(q_1 \rightarrow ((\neg(B))Uq_3))$, $G(F(q_1)), G(\neg B)$ |
| Multiple choice | $F(G(Z_{target}))$ |
| Navigation | $G(F(Z_0))$ |
| Non-linear circular | $G(F(R_{0,\alpha})), G(\neg B)$ |
| Multilayer control lin | $G(q_i \rightarrow (\neg q_J W q_K)) \rightarrow (G(\neg B)), G(G(q_1) \rightarrow F(G(Z_0)))$ |
| Multilayer control nonlin | $G(q_i \rightarrow (\neg q_J W q_K)) \rightarrow (G(\neg B)), G(G(q_1) \rightarrow F(G(Z_0)))$ |
| Bball | $G(F(v \leq 1))$ |
| Circle | $G(Z_0 \rightarrow F(Z_1))$ |
| Drivetrain | $G(q_1 \wedge (F\neg(q_2)) \rightarrow F(\neg(q_3)))$ |
| Rendezvous | $F(G(q_{final}))$ |

Table 2: Properties proved by VeriLHyS framework

As the general problem of LTL verification for hybrid systems is undecidable, the VeriLHyS framework is not guaranteed to terminate successfully for all inputs and in many cases it may instead yield an "Unknown" outcome or time out.

Such unresolved cases may arise, for example, when the RSO computation produces very coarse bounds due to high-dimensional state space or prolonged time horizons, leading to ambiguous LTL constraints. Clearly, other motivations are possible.

The complete set of results, including the cases that currently return "Unknown" or time out, are publicly available on the project website as open challenges.

# 5 Conclusions

In this paper, we proposed an extension of the SpaceEx format to specify LTL properties and an associated suite of benchmarks. This suite, which hopefully will be in the future enlarged with other contributions, already includes a variety of models, ranging from low-dimensional linear systems to more complex nonlinear and scalable instances. The new format is supported by the tool VeriLHyS which can be used as baseline for a comparison on the benchmarks. In the future, the format can be further extended for example to include also metric (MTL/STL) temporal operators.

# References

[Alt15] Matthias Althoff. An introduction to CORA 2015. In *Proc. of the 1st and 2nd Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151. EasyChair, December 2015.

[BD17] Stanley Bak and Parasara Sridhar Duggirala. HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems. In *HSCC*, pages 173–178. ACM, 2017.

[BT25] Ludovico Battista and Stefano Tonetta. Deriving Liveness Properties of Hybrid Systems from Reachable Sets and Lyapunov-Like Certificates. In Meenakshi D'Souza, Raghavan Komondoor, and B. Srivathsan, editors, *Automated Technology for Verification and Analysis - 23rd International Symposium, ATVA 2025, Bengaluru, India, October 27-31, 2025, Proceedings*, Lecture Notes in Computer Science, pages 363–386. Springer, 2025.

[BTZ] Ludovico Battista, Stefano Tonetta, and Gianni Zampedri. Verilhys: a framework for ltl specification and verification of hybrid systems. To appear in TACAS26, available at https://es-static.fbk.eu/people/lbattista/pages/verilhys_tool.html.

[CÁS13] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An Analyzer for Non-linear Hybrid Systems. In *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 258–263. Springer, 2013.

[CRT09] Alessandro Cimatti, Marco Roveri, and Stefano Tonetta. Requirements validation for hybrid systems. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification*, pages 188–203, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[dAM95] Luca de Alfaro and Zohar Manna. Verification in continuous time by discrete reasoning. In V. S. Alagar and Maurice Nivat, editors, *Algebraic Methodology and Software Technology*, pages 292–306, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[FGD+11] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable Verification of Hybrid Systems. In *CAV*, volume 6806 of *Lecture Notes in Computer Science*, pages 379–395. Springer, 2011.

[Hen96] T.A. Henzinger. The theory of hybrid automata. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292, 1996.

[SÁMK17] Stefan Schupp, Erika Ábrahám, Ibtissem Ben Makhlouf, and Stefan Kowalewski. HyPro: A C++ Library of State Set Representations for Hybrid Systems Reachability Analysis. In *NFM*, volume 10227 of *Lecture Notes in Computer Science*, pages 288–294, 2017.

# A Syntax and Semantics of LTL over Hybrid Automata

For the reader's convenience, this appendix provides the formal definitions of hybrid automata (the standard interpretation of the SpaceEx models) and the corresponding semantics of LTL on such hybrid systems, providing the interpretation of the LTL extension of the SpaceEx format. These definitions are taken from [BT25], which in turn largely derived them from the literature, mainly from [Hen96] and [CRT09].

## A.1 Hybrid Automata

**Definition 1.** *A* Hybrid Automaton *(HA) is a tuple* $H = (X, Q, q_{init}, E, init, flow, inv, guard, jump)$*, where:*

- $X$ *is a finite set of real variables* $x_1, \ldots, x_n$*;*
- $Q$ *is a finite set of discrete modes (also called locations);*
- $q_{init} \in Q$ *is the initial location;*
- $E \subseteq Q \times Q$ *is the set of possible discrete transitions;*
- $init \subseteq \mathbb{R}^n$ *specifies the set of initial continuous states;*
- $inv\colon Q \to 2^{\mathbb{R}^n}$ *assigns an invariant set* $inv(q)$ *to each location* $q$*, that specifies where the system is allowed to remain while in mode* $q$*;*
- $flow\colon Q \times \mathbb{R}^n \to \mathbb{R}^n$ *is a partial function that defines the continuous evolution of the system in each discrete mode* $q \in Q$ *via a set of differential equations;*
- $guard\colon E \to 2^{\mathbb{R}^n}$ *is a guard condition, specifying when the transition* $(q, q')$ *is enabled based on the continuous state;*
- $jump\colon E \to 2^{\mathbb{R}^n \times \mathbb{R}^n}$ *assigns a relation to each edge that specifies the possible* jumps *of the continuous variables.*

A state of $H$ is a pair $\langle q, s\rangle$, where $q \in Q$ and $s$ is an assignment to the variables in $X$; $s$ is also called a continuous state.

We consider a symbolic representation of the conditions $init, inv(q), flow(q, s)$, $guard(e)$, and $jump(e)$, in the sense that they are defined by symbolic formulas over the variables $X$, taken from a set denoted by $Expr(X)$ (or $Expr(X, X')$ in the case of $jump$). In this work, we focus on *polynomial automata*, where $Expr(X)$ contains Boolean combinations of polynomial constraints over $X$. This is mainly due to limitations of theory solvers and simulators, but the proposed method can be generalized to more general forms. We also assume, for simplicity, that $E$ does not contain elements of the form $(q, q)$.

To describe the evolution of a hybrid system $H$ we will use the notion of *hybrid trace* [dAM95, CRT09]. We briefly recall it. Given an interval $I \subseteq \mathbb{R}$, we denote by $l(I)$ its infimum and by $u(I)$ its supremum.

**Definition 2.** *A hybrid trace for* $H$ *is an infinite sequence* $\sigma = \langle f_0, I_0, q_0\rangle, \langle f_1, I_1, q_1\rangle, \ldots$ *such that:* $I_i$ *are adjacent intervals,* i.e. $l(I_{i+1}) = u(I_i)$*;* $q_i$ *are locations,* i.e. $q_i \in Q$*; the intervals cover* $\mathbb{R}^{\geq 0}$*,* i.e. $\bigcup_{i\in\mathbb{N}} I_i = \mathbb{R}^{\geq 0}$*;* $f_i\colon I_i \to \mathbb{R}^n$ *is analytic. We sometimes denote* $q_i$ *with* $loc(\sigma_i)$*.*

**Definition 3.** *A hybrid trace is a* trajectory *(also called a* run *or a* solution*) for* $H$ *if:*

- *the image of* $f_i$ *is contained in* $inv(q_i)$.
- *if* $l(I_i) \neq u(I_i)$, *then* $\frac{df_i}{dt}(t) = flow(q_i)(f_i(t))$ *for all* $t \in I_i$;
- *if* $q_i = q_{i+1}$, *then* $f_i \cup f_{i+1}$ *is well defined and analytic on* $I_i \cup I_{i+1}$. *In particular, either* $I_i$ *is right-closed or* $I_{i+1}$ *is left-closed;*
- *if* $q_i \neq q_{i+1}$, *then* $I_i$ *is right-closed,* $I_{i+1}$ *is left-closed,* $e = (q_i, q_{i+1}) \in E$, $f_i(u(I_i)) \in guard(e)$, *and* $(f_i(u(I_i)), f_{i+1}(l(I_{i+1}))) \models jump(e)$.

We say that a run *starts in* $(R, q)$ if $f_0(0) \in R$ and $loc(\sigma_0) = q$. An *initial run* is a run that starts in $(init, q_{init})$. We notice that different traces may describe the same dynamics. Intuitively, a trace is a sampling refinement of another one if it has been obtained by splitting an interval into two parts [dAM95].

**Definition 4** (Partitioning Function - Sampling Refinement [dAM95])**.** *A partitioning function* $\mu$ *is a sequence* $\mu_0, \mu_1, \mu_2, \ldots$ *of non-empty, adjacent and disjoint intervals of* $\mathbb{N}$ *partitioning* $\mathbb{N}$. *Formally,* $\bigcup_{i\in\mathbb{N}} \mu_i = \mathbb{N}$ *and* $u(\mu_i) = l(\mu_{i+1}) - 1$.

*Given two hybrid traces* $\sigma = \langle f_0, I_0, q_0\rangle, \langle f_1, I_1, q_1\rangle, \ldots$ *and* $\sigma' = \langle f'_0, I'_0, q'_0\rangle, \langle f'_1, I'_1, q'_1\rangle, \ldots$, *we say that* $\sigma'$ *is a sampling refinement of* $\sigma$ *by the partitioning* $\mu$ *(denoted by* $\sigma' \preceq_\mu \sigma$*) iff, for all* $i \in \mathbb{N}$, $I_i = \bigcup_{j\in\mu_i} I'_j$, *and for all* $j \in \mu_i$, $f'_j = f_i$ *and* $q'_j = q_i$.

## A.2 LTL

We extend LTL replacing propositions with predicates over a set $Q$ of locations and a set $X$ of real variables. The syntax of LTL formulas is given by the following grammar rules: $\phi := q \mid R \mid \phi \vee \phi \mid \neg\phi \mid \phi U \phi$, where $q \in Q$ and $R$ ranges over $Expr(X)$. We use the same abbreviations defined for LTL.

Given a hybrid trace $\sigma = \langle f_0, I_0, q_0\rangle, \langle f_1, I_1, q_1\rangle, \ldots$ and an LTL formula $\phi$ over $Q$ and $X$, we define $\sigma \models \phi$ as follows:

- $\sigma, i \models q$ iff $q_i = q$;
- $\sigma, i \models R$ iff for all $t \in I_i$, $f_i(t) \models R$;
- $\sigma, i \models \phi \vee \psi$ iff $\sigma, i \models \phi$ or $\sigma, i \models \psi$
- $\sigma, i \models \neg\phi$ iff $\sigma, i \not\models \phi$
- $\sigma, i \models \phi U \psi$ iff there exists $k \geq i$ s.t. $\sigma, k \models \psi$ and for all $j$, $i \leq j < k$, $\sigma, j \models \phi$

Finally, $\sigma \models \phi$ iff $\sigma, 0 \models \phi$. We say that a hybrid trace $\sigma$ is ground for a a predicate $R$ if its interpretation over intervals is constant, *i.e.*, $\forall(t_1, t_2) \in I_i$, $(f_i(t_1) \models R \Leftrightarrow f_i(t_2) \models R)$. We say that $\sigma$ is ground for an LTL formula $\phi$ iff it is ground for every predicate in $\phi$. Given a hybrid automaton $H$ and an LTL formula $\phi$ over $Q$ and $X$, $H \models \phi$ iff, for all initial runs $\sigma$ of $H$ that are ground for $\phi$, $\sigma \models \phi$. Notice that it is always possible to refine a run to make it ground for a LTL formula [dAM95, CRT09].