Linear Temporal Logic for Hyperproperties (HyperLTL)

Course: Specification and Verification of Parallel Systems

29 November 2019

Presented by: Elahe Fazeldehkordi

UiO : Universitetet i Oslo

Hyperproperties

- Trace: a sequence of states
- System: is modeled by a non-empty set of infinite traces, called its executions
- Trace property: a set of infinite traces

If systems are modeled as sets of execution traces, then the extension of a system property is a set of sets of infinite traces or, equivalently, a set of trace properties. This type of set is named a **hyperproperty**.

Every property of system behavior (for systems modeled as trace sets) can be specified as a hyperproperty.

Important security policies cannot be expressed as properties of individual execution traces of a system.

- whether a trace is allowed by the policy depends on whether another trace is also allowed

Hyperproperties can describe:

- trace properties
- security policies, such as:
  - noninterference
  - mean response time

HyperLTL

- By Clarkson et al. 2014 is an extension of LTL for specifying hyperproperties.
- Generalizes linear-time temporal logic (LTL)
- Examines more than one execution trace at a time
- Allows explicit quantification over multiple execution traces simultaneously
- Allows propositions that stipulate relationships among those traces
- Provides a simple and unifying logic in which many information-flow security policies can be directly expressed

LTL and HyperLTL

- Trace properties are typically specified in temporal logics, most prominently in Linear Temporal Logic (LTL).
- Verification of LTL specifications is routinely employed in industrial settings and marks one of the most successful applications of formal methods to real-life problems.
- LTL implicitly quantifies over only a single path at a time, hence cannot express many hyperproperties of interest.
- In LTL the satisfying object is a trace. Syntax:

$$\varphi \;::=\; a \;\mid\; \neg\varphi \;\mid\; \varphi \vee \varphi \;\mid\; \mathrm{X}\,\varphi \;\mid\; \varphi\,\mathrm{U}\,\varphi$$

- In HyperLTL the satisfying object is a set of traces and a trace assignment:

$$\begin{aligned} \psi \;&::=\; \exists\pi.\,\psi \;\mid\; \forall\pi.\,\psi \;\mid\; \varphi \\ \varphi \;&::=\; a_\pi \;\mid\; \neg\varphi \;\mid\; \varphi \vee \varphi \;\mid\; \mathrm{X}\,\varphi \;\mid\; \varphi\,\mathrm{U}\,\varphi \end{aligned}$$

Syntax

Formulas of HyperLTL are defined by the following grammar:

$$\begin{array}{llllllllll}\psi & ::= & \exists\pi.\ \psi & \mid & \forall\pi.\ \psi & \mid & \varphi & & & \\ \varphi & ::= & a_\pi & \mid & \neg\varphi & \mid & \varphi \vee \varphi & \mid & \mathrm{X}\,\varphi & \mid & \varphi\,\mathrm{U}\,\varphi\end{array}$$

π is a trace variable from an infinite supply $\mathcal{V}$ of trace variables.

$\forall\pi_1.\forall\pi_2.\exists\pi_3.\psi$ means that for all traces π_1 and π_2, there exists another trace π_3, such that ψ holds on those three traces.

X_φ means that φ holds on the next state of every quantified trace.

Syntax

- Implication: $\varphi_1 \rightarrow \varphi_2 \equiv \neg \varphi_1 \vee \varphi_2$
- Conjunction: $\varphi_1 \wedge \varphi_2 \equiv \neg (\neg \varphi_1 \vee \neg \varphi_2)$
- Bi-implication: $\varphi_1 \leftrightarrow \varphi_2 \equiv (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$
- True and false: $a_\pi \vee \neg a_\pi$ and $\neg\, true$
- Other standard temporal connectives are:
  - $F\varphi \equiv true\ U\ \varphi$
  - $G\varphi \equiv \neg F \neg \varphi$
  - $\varphi_1\, W\, \varphi_2 \equiv (\varphi_1\, U \varphi_2) \vee G\varphi_1$
  - $\varphi_1\ R\ \varphi_2 \equiv \neg (\neg \varphi_1\, U\, \neg\varphi_2)$
- $\varphi_1 \cup \varphi_2$ means that φ_2 will eventually hold of the states of all quantified traces that appear at the same index, and until then φ_1 holds.

Semantics

Validity:

$$
\begin{array}{lll}
\Pi \models_T \exists \pi.\, \psi & \text{iff} & \text{there exists } t \in T : \Pi[\pi \mapsto t] \models_T \psi \\
\Pi \models_T \forall \pi.\, \psi & \text{iff} & \text{for all } t \in T : \Pi[\pi \mapsto t] \models_T \psi \\
\Pi \models_T a_\pi & \text{iff} & a \in \Pi(\pi)[0] \\
\Pi \models_T \neg\varphi & \text{iff} & \Pi \not\models_T \varphi \\
\Pi \models_T \varphi_1 \vee \varphi_2 & \text{iff} & \Pi \models_T \varphi_1 \text{ or } \Pi \models_T \varphi_2 \\
\Pi \models_T \mathrm{X}\,\varphi & \text{iff} & \Pi[1,\infty] \models_T \varphi \\
\Pi \models_T \varphi_1 \,\mathrm{U}\, \varphi_2 & \text{iff} & \text{there exists } i \geq 0 : \Pi[i,\infty] \models_T \varphi_2 \\
 & & \quad \text{and for all } 0 \leq j < i \text{ we have } \Pi[j,\infty] \models_T \varphi_1
\end{array}
$$

- Trace assignment suffix $\Pi[i,\infty]$ denotes the trace assignment below for all π
  $$\Pi'(\pi) = \Pi(\pi)[i,\infty]$$

- If $\Pi \models_T \varphi$ holds for the empty assignment Π, then T satisfies φ .

Semantics

- A Kripke structure K is a tuple (S, s_0, δ, AP, L)
  - a set of states S ,
  - an initial state $s_0 \in S$,
  - a transition function δ
  - $S \rightarrow 2^S$, a set of atomic propositions AP
  - a labeling function $L : S \rightarrow 2^{AP}$.

- To ensure that all traces are infinite, we require that $\delta(s)$ is nonempty for every state s .
- The set Traces (K) of traces of K is the set of all sequences of labels produced by the state transitions of K starting from initial state.
- Traces (K) contains trace t iff there exists a sequence $s_0 s_1 \ldots$ of states, such that s_0 is the initial state, and for all $i \geq 0$, it holds that $s_i + 1 \in \delta(s_i)$; and $t[i] = L(s_i)$.
- A Kripke structure K satisfies φ , denoted by $K \vDash \varphi$, if Traces (K) satisfies φ .

Security Policies in HyperLTL

- **Noninterference:** the outputs observed by low-security users are the same as they would be in the absence of inputs submitted by high-security users.

- Noninference is a variant of noninterference.

- **Noninference:** for all traces, the low-observable behavior must not change when all high inputs are replaced by a dummy input λ , that is, when the high input is removed.

Noninference in HyperLTL:

$$\forall \pi . \exists \pi' . (G\ \lambda_{\pi'}) \land \pi =_L \pi'$$

$\lambda_{\pi'}$ expresses that all of the high inputs in the current state of π' are λ,
$\pi =_L \pi'$ expresses that all low variables in π and π' have the same values.

Security Policies in HyperLTL

- A (nondeterministic) program satisfies **observational determinism** if every pair of traces with the same initial low observation remain indistinguishable for low users.

Observational determinism in HyperLTL:

$$\forall \pi. \forall \pi'. \pi[0] =_{L,in} \pi'[0] \rightarrow \pi =_{L,out} \pi'$$

Where $\pi =_{L,in} \pi'$ and $\pi =_{L,out} \pi'$ express that both traces agree on the low input and low output variables, respectively.

Problems about HyperLTL:

- Bounded termination is not expressible.
- Satisfiability problem is undecidable.

References

1. Clarkson, Michael R., et al. "Temporal logics for hyperproperties." *International Conference on Principles of Security and Trust*. Springer, Berlin, Heidelberg, 2014.

2. Clarkson, Michael R., and Fred B. Schneider. "Hyperproperties." *Journal of Computer Security* 18.6 (2010): 1157-1210.

3. Goguen, Joseph A., and José Meseguer. "Security policies and security models." *1982 IEEE Symposium on Security and Privacy*. IEEE, 1982.