Hyper Separation Logic: (Dis-)Proving Hyperproperties of Programs with Pointers

Master's Thesis Project Description

Trayan Gospodinov
supervised by
Thibault Dardinier, Prof. Dr. Peter Müller and Prof. Dr. Tinko Tinchev

November 29, 2023

1 Introduction

Throughout the years, many Hoare logics have been developed for proving trace properties of computer programs. Trace properties, as the name suggests, concern only individual program executions, e.g. functional correctness. However, many program properties of interest concern more than one program execution. Such properties are called hyperproperties [Clarkson and Schneider 2008], e.g. determinism (executing the program twice in the same initial state results in the same final state) and non-interference [Volpano et al. 1996] (executing the program twice with the same low-sensitivity inputs results in the same low-sensitivity outputs). Some special cases, such as 2-safety hyperproperties[1], have been extensively studied, whereas logics specifically designed for handling broader range of hyperproperties are rather scarce.

The formulae in Hoare logics [Hoare 1969] are the so called Hoare triples $\{P\}C\{Q\}$, where P (precondition) and Q (postcondition) are assertions and C is a computer program. A Hoare triple is valid iff executing C in a program state, satisfying P, results in a program state satisfying Q. Hoare logic is an overapproximating logic, because the set of states satisfying Q is a superset of the set of reachable states, starting in a state satisfying P and executing C, i.e. $\{\sigma \in \text{States} : Q(\sigma)\} \supseteq \{\sigma' : \exists \sigma \in \text{States}.\, P(\sigma) \land \langle C, \sigma\rangle \to \sigma'\}$[2]. Similarly, we can define underapproximating Hoare triples $[P]C[Q]$, valid, iff for every state σ', satisfying Q, there is a state σ satisfying P, s.t. $\langle C, \sigma\rangle \to \sigma'$, i.e. valid iff $\{\sigma \in \text{States} : Q(\sigma)\} \subseteq \{\sigma' : \exists \sigma \in \text{States}.\, P(\sigma) \land \langle C, \sigma\rangle \to \sigma'\}$. Such underapproximating Hoare logics have been well developed, e.g. Reverse Hoare logic [de Vries and Koutavas 2011] and Incorrectness logic [O'Hearn 2019].

[1] Hyperproperties concerning 2 program executions, stating that "nothing bad will happen".
[2] Here we use big-step semantic notation $\langle C, \sigma\rangle \to \sigma'$.

Earlier, we pointed out that extensive research in the field, concerning special cases of hyperproperties, has also been conducted, e.g. Relational Hoare logic [Benton 2004] and Cartesian Hoare Logic [Sousa and Dillig 2016]. Now, we shift our focus on a novel logic, that can reason about arbitrary hyperproperties over terminating executions, called Hyper Hoare logic [Dardinier and Müller 2023]. The formulae of Hyper Hoare logic are the so called hyper-triples $[P]C[Q]$, where P and Q are assertions over sets of states and C is a computer program. Such a hyper-triple is valid iff for any set of initial states S that satisfies P, the set of all final states that can be reached by executing C in some state from S satisfies the postcondition Q.

Consider the hyperproperty *non-interference*, which holds iff $\forall \sigma_1, \sigma_2, \sigma_1', \sigma_2' \in \text{States}.\, \sigma_1 \upharpoonright L = \sigma_2 \upharpoonright L \Rightarrow \langle C, \sigma_1\rangle \to \sigma_1' \Rightarrow \langle C, \sigma_2\rangle \to \sigma_2' \Rightarrow \sigma_1' \upharpoonright L = \sigma_2' \upharpoonright L$[3,4], where states are functions from program variables $PVAR$ to values and $L \subseteq PVAR$ is the set of low-sensitivity variables. Assuming we have only one low variable l for simplicity, i.e. $L = \{l\}$, a hyper-triple that ensures $\text{NI}_L(C)$ looks like this: $[\forall\langle\sigma_1\rangle, \langle\sigma_2\rangle.\, \sigma_1(l) = \sigma_2(l)]C[\forall\langle\sigma_1'\rangle, \langle\sigma_2'\rangle.\, \sigma_1'(l) = \sigma_2'(l)]$, where the precondition $\forall\langle\sigma_1\rangle, \langle\sigma_2\rangle.\, \sigma_1(l) = \sigma_2(l)$ desugars[5] to $\lambda S.\, \forall \sigma_1, \sigma_2 \in S.\, \sigma_1(l) = \sigma_2(l)$ and the postcondition desugars to $\lambda S'.\, \forall \sigma_1', \sigma_2' \in S'.\, \sigma_1'(l) = \sigma_2'(l)$. This hyperproperty has been generalized for non-deterministic programs, since the former formulation is too restrictive for non-deterministic programs, e.g. the following program $C \triangleq (l := h + \mathit{randUnboundedInt}())$ is information flow secure, but $\text{NI}_{\{l\}}(C)$ does not hold. The generalization is often formalized as *generalized non-interference* [McCullough 1987; Mclean 1996], where $\text{GNI}_L(C)$ holds iff $\forall \sigma_1, \sigma_2, \sigma_1', \sigma_2' \in \text{States}.\, \exists \sigma' \in \text{States}.\, \sigma_1 \upharpoonright L = \sigma_2 \upharpoonright L \Rightarrow \langle C, \sigma_1\rangle \to \sigma_1' \Rightarrow \langle C, \sigma_2\rangle \to \sigma_2' \Rightarrow \sigma_1' \upharpoonright L \neq \sigma_2' \upharpoonright L \Rightarrow \langle C, \sigma_1\rangle \to \sigma' \wedge \sigma' \upharpoonright L = \sigma_2' \upharpoonright L$. Assuming we have only one low variable l, i.e. $L = \{l\}$, and that C does not modify variable h for simplicity, a hyper-triple that ensures $\text{GNI}_L(C)$ looks like this: $[\forall\langle\sigma_1\rangle, \langle\sigma_2\rangle.\, \sigma_1(l) = \sigma_2(l)]C[\forall\langle\sigma_1'\rangle, \langle\sigma_2'\rangle.\, \exists\langle\sigma'\rangle.\, \sigma'(h) = \sigma_1'(h) \wedge \sigma'(l) = \sigma_2'(l)]$. To the best of our knowledge, Hyper Hoare logic is the only Hoare logic that can simultaneously prove GNI_L ($\forall\exists$-hyperproperty) and disprove GNI_L ($\exists\forall$-hyperproperty) for arbitrary L.

The above mentioned Hoare logics cannot reason about pointer programs without being cumbersome at best. A more elegant approach, based on *local reasoning*, has been developed, called Separation logic [Reynolds 2002]. Most, if not all, of the more well-established Hoare logics, have been successfully extended to support heap operations, based on the key features of Separation logic, e.g. Incorrectness Separation logic [Raad et al. 2020] and Relational Separation logic [Yang 2007].

[3]Here $f \upharpoonright A$ is the restriction of function f to the set A, i.e. $f \upharpoonright A = f \cap (A \times \text{Rng}(f))$.
[4]We consider implication to be right-associative.
[5]Recall that we defined hyper-triples' assertions to be over set of states.

2 Approach

Hyper Hoare logic has been shown to be sound, complete and has been demonstrated to capture and go beyond the properties supported by numerous existing correctness and incorrectness logics. Nevertheless, said logic cannot reason about pointer programs, which makes it inapplicable for the majority of the real-world scenarios. Thankfully, history has shown that, in general, Hoare logics can be extended to support heap operations, drawing upon the fundamental principles of Separation logic.

3 Goals

3.1 Core Goals

Considering the above mentioned Separation logics, clear goals emerge for our pursuit of extending Hyper Hoare logic to Hyper Separation logic:

- Add a heap to the state model and its core operations to the programming language: cons, lookup, update and free. Define small-step semantics for our programming language;
- Add "points to" to the assertion language and develop rules based on semantic assertions for cons, lookup, update and free;
- Define separation conjunction between hyper-assertions and prove a frame rule sound;
- Apply the logic on interesting examples, come up with new ones;
- Compare with Relational Separation logic and Outcome logic;
- Formalize everything in Isabelle/HOL [Nipkow et al. 2002].

The primary challenge in achieving the core goals of the project would be to define separating conjunction in an elegant way and to provide a sound frame rule.

3.2 Extension Goals

- Explore an extension of the logic to handle parallelism
  - Add parallel composition and atomic blocks to the programming language;
  - Explore the soundness of the Par rule;
  - Explore the soundness of invariants with atomic blocks;
- Explore a relational extension of HHL (with multiple programs).

References

Nick Benton. Simple relational correctness proofs for static analyses and program transformations. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '04, page 14–25, New York, NY, USA, 2004. Association for Computing Machinery. URL https://doi.org/10.1145/964001.964003.

Michael R. Clarkson and Fred B. Schneider. Hyperproperties. In *2008 21st IEEE Computer Security Foundations Symposium*, pages 51–65, 2008. URL https://doi.org/10.1109/CSF.2008.7.

Thibault Dardinier and Peter Müller. Hyper hoare logic: (dis-)proving program hyperproperties (extended version), 2023.

Edsko de Vries and Vasileios Koutavas. Reverse hoare logic. In Gilles Barthe, Alberto Pardo, and Gerardo Schneider, editors, *Software Engineering and Formal Methods*, pages 155–171, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, oct 1969. URL https://doi.org/10.1145/363235.363259.

Daryl McCullough. Specifications for multi-level security and a hook-up. In *1987 IEEE Symposium on Security and Privacy*, pages 161–161, 1987. URL https://doi.org/10.1109/SP.1987.10009.

John Mclean. A general theory of composition for a class of "possibilistic" properties. *IEEE Transactions on Software Engineering*, 22(1):53 – 67, 02 1996. doi: 10.1109/32.481534. URL https://doi.org/10.1109/32.481534.

Tobias Nipkow, Markus Wenzel, and Lawrence C. Paulson. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer-Verlag, Berlin, Heidelberg, 2002.

Peter W. O'Hearn. Incorrectness logic. *Proc. ACM Program. Lang.*, 4(POPL): 1–32, dec 2019. URL https://doi.org/10.1145/3371078.

Azalea Raad, Josh Berdine, Hoang-Hai Dang, Derek Dreyer, Peter O'Hearn, and Jules Villard. *Local Reasoning About the Presence of Bugs: Incorrectness Separation Logic*, pages 225–252. Springer International Publishing, 07 2020. URL https://doi.org/10.1007/978-3-030-53291-8_14.

J.C. Reynolds. Separation logic: a logic for shared mutable data structures. In *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*, pages 55–74, 2002. URL https://doi.org/10.1109/LICS.2002.1029817.

Marcelo Sousa and Isil Dillig. Cartesian hoare logic for verifying k-safety properties. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '16, page 57–69,

New York, NY, USA, 2016. Association for Computing Machinery. URL https://doi.org/10.1145/2908080.2908092.

Dennis Volpano, Cynthia Irvine, and Geoffrey Smith. A sound type system for secure flow analysis. *J. Comput. Secur.*, 4(2–3):167–187, jan 1996.

Hongseok Yang. Relational separation logic. *Theoretical Computer Science*, 375 (1):308–334, 2007. URL https://doi.org/10.1016/j.tcs.2006.12.036.