A Refined Proof Theory for Reasoning About Separation

Limin Jia David Walker
Princeton University

E-mail: {ljia,dpw}@cs.princeton.edu

Background. Automated program verification tools can help programmers to discover bugs at an early stage, thus producing more reliable software. A large number of software bugs are related to operations on shared mutable data structures such as memory allocation, deallocation, updating the content of pointers. Therefore, being able to reason about the correctness and safety properties of programs that manipulate shared mutable data structures is essential to producing high-assurance software.

The recent development of separation logic [1, 2] by Reynolds, O'Hearn, and Yang has shown that using substructural logic to describe memory as disjoint pieces is a promising approach to verify the correctness of programs that manipulate list data structures. Now the challenge is to develop theorem provers for separation logic so that programmers can write high-assurance software without the burden of constructing all of their correctness proofs by hand.

Some Simple Observations. Reynolds, O'Hearn and Yang's separation logic is a complex system based on the logic of bunched implications. Our hope is that by rethinking the proof theory necessary for reasoning about program data we will simplify the theorem-proving tasking and will be able to bring off-the-shelf theorem proving components to bare on the project.

One of the things that makes bunched implications more complex than other logics is the presence of its eponymous contexts — the "bunches." Fortunately, it appears that complex bunches appear infrequently, if at all, in proofs about program data structures. Consequently, we believe we can develop a very effective tool for reasoning about program data based on linear logic, which shares all the connectives of bunched logic except the implication $\rightarrow$, which is replaced by the modality !.

Linear logic also has the advantage that it directly addresses one of the somewhat ad hoc concepts in separation logic – the heap-free formulas. Heap-free formulas, such arithmetic equations, do not depend upon the store and are always subject to contraction, weakening and exchange. A natural way to deal with these predicates is to treat them as ordinary formulas but wrap them with unrestricted linear logical modality !. By doing so, we may exploit existing theorem proving techniques for dealing with ! rather than having to come up with our own new techniques. We also use ! to help us represent aliased datastructures concisely, which separation logic appears to have great difficulty with.

Finally, we have observed that while bunched implications is classical to the core, the classical reasoning occurs at the level of arithmetic as opposed to at the level of the store. Reasoning about store and its transformations can be achieved effectively in an intuitionistic logic.

ILC. We developed a new logic named ILC (Intuitionistic Linear logic with Classical arithmetic) that is designed as a foundation for automatic reasoning about state and takes the above observations into account. The basic goal is to embed classical reasoning about arithmetic into intuitionistic linear logic. We do so in a principled way by confining the classical formulas to a new modality $\bigcirc$. In addition, our logic contains all the connectives of first-order multiplicative and additive linear logic. As in O'Hearn et al.'s work, the multiplicative conjunction $\otimes$ is used to describe the disjoint of two stores, the multiplicative implication $\multimap$ is used to describe store updates and the additive & is used to describe sharing. Linear logic's unrestricted modality ! is used to describe heap-free data, and aliased data similar to that in previous work on alias types [3]. We use E to range over integer expressions; Pa ranges over arithmetic predicates, which are equality and partial-order relationship on integers. The classical formulas A consist of classical truth, arithmetic predicates, conjunction and negation. The first store predicate $(E_1 \Rightarrow E_2)$ describes a heap containing only one location, E_1 and contents is E_2, while the second store predicate $(E_1 \overset{a}{\Rightarrow} E_2)$ describes possibly aliased data. Disjunction and universal and existential quantifiers are not shown.

Int Exps	E	$::=$	$n \mid xi \mid E + E \mid -E$
Arith. Preds	Pa	$::=$	$E_1 = E_2 \mid E_1 < E_2$
Classic. Forms	A	$::=$	$true \mid Pa \mid A_1 \wedge A_2 \mid \neg A$
Store Preds	P_s	$::=$	$(E_1 \Rightarrow E_2) \mid (E_1 \overset{a}{\Rightarrow} E_2)$
Intuit. Forms	F	$::=$	$P_s \mid \mathbf{1} \mid F_1 \otimes F_2 \mid F_1 \multimap F_2$
			$\mid \top \mid F_1 \,\&\, F_2 \mid !\,F \mid \bigcirc A$

Logical Judgments. The hypothetical judgments of our logic have the form of $\Gamma\,;\,\Theta\,;\,\Delta \implies F$. This judgement includes an unrestricted context Γ for classical formulas, an unrestricted context Θ for intuitionistic formulas, and a linear context Δ, also for intuitionistic formulas. The first two contexts have contraction, weakening and exchange properties, while the last has only exchange.

An intuitive reading of the sequent $\Gamma\,;\,\Theta\,;\,\Delta \implies F$ is that given a state described by unrestricted assumptions Θ, linear assumptions Δ, and satisfying all the classical arithmetic constraints in Γ, this state can also be viewed as a state described by F.

Sequent Rules. For the classical part of our logic, we resort to Gentzen's LK formalization. The sequent rules for classical logic are in the form: $\Gamma \# \Gamma'$, where Γ is the context for truth assumptions and Γ' is the context for false assumptions. The sequent $\Gamma \# \Gamma'$ can be read as: the truth assumption Γ contradicts with one of the false assumptions in Γ'. The formalization is standard and we only give the basic *Contra* rule below.

$$\frac{}{\Gamma, A \# A, \Gamma'}\ \textit{Contra}$$

The sequent rules for most connectives are the same as those in intuitionistic linear logic, except that the classical context Γ is carried around. The interesting rules are the left and right rule for the new modality $\bigcirc$, and the *absurdity* rule, which illustrates the interaction between the classical part and the intuitionistic part of the logic. The right rule for $\bigcirc$ says that if Γ contradicts the assertion "A false" (which means A is true) then we can derive $\bigcirc A$, without using any linear resources. If we read the left rule for $\bigcirc$ bottom up, it says that whenever we have $\bigcirc A$, we can put A together with other classical assumptions in Γ. The absurdity rule is a peculiar one. The justification for this rule is that since Γ is not consistent, no state can meet the constraints imposed by Γ, and therefore, any statement based on the assumption that a state satisfies those constraints is simply true.

$$\frac{\Gamma \# A}{\Gamma\,;\,\Theta\,;\,\cdot \implies \bigcirc A}\ \bigcirc R \qquad \frac{\Gamma, A\,;\,\Theta\,;\,\Delta \implies F}{\Gamma\,;\,\Theta\,;\,\Delta, \bigcirc A \implies F}\ \bigcirc L$$

$$\frac{\Gamma \# \cdot}{\Gamma\,;\,\Theta\,;\,\Delta \implies F}\ \textit{Absurdity}$$

$$\frac{\Gamma\,;\,\Theta\,;\,\Delta_1 \implies F_1 \quad \Gamma\,;\,\Theta\,;\,\Delta_2 \implies F_2}{\Gamma\,;\,\Theta\,;\,\Delta_1, \Delta_2 \implies F_1 \otimes F_2}\ \otimes R$$

$$\frac{\Gamma\,;\,\Theta\,;\,\Delta, F_1, F_2 \implies F}{\Gamma\,;\,\Theta\,;\,\Delta, F_1 \otimes F_2 \implies F}\ \otimes L$$

We have proven the following cut elimination theorem for our logic.

Theorem 1 (Cut Elimination)

1. *If $\Gamma, A \# \Gamma'$ and $\Gamma \# A, \Gamma'$ then $\Gamma \# \Gamma'$.*
2. *If $\Gamma \# A$ and $\Gamma, A\,;\,\Theta\,;\,\Delta \implies F$ then $\Gamma\,;\,\Theta\,;\,\Delta \implies F$.*
3. *If $\Gamma\,;\,\Theta\,;\,\cdot \implies F$ and $\Gamma\,;\,\Theta, F\,;\,\Delta \implies F'$ then $\Gamma\,;\,\Theta\,;\,\Delta \implies F'$.*
4. *If $\Gamma\,;\,\Theta\,;\,\Delta \implies F$ and $\Gamma\,;\,\Theta\,;\,\Delta', F \implies F'$ then $\Gamma\,;\,\Theta\,;\,\Delta, \Delta' \implies F'$.*

Continuing research. We are currently developing a Hoare Logic that uses ILC as its assertion language. The Hoare Logic is a syntax-directed set of verification-condition generation rules. We are planning on building an automated theorem prover to discharge the verification conditions. One additional property of our logic that increases our chance of success is that all the classical reasoning can be pushed to the leaves of the derivation tree. Consequently, we believe we will be able to build our theorem prover rather directly by combining theorem proving techniques for linear logic with classical decision procedures for arithmetic.

Acknowledgement. We would like to thank Frank Pfenning for fruitful discussions about this research and for helping us work out the reading of our sequents.

References

[1] S. Ishtiaq and P. O'Hearn. BI as an assertion language for mutable data structures. In *Twenty-Eighth ACM Symposium on Principles of Programming Languages*, pages 14–26, London, UK, Jan. 2001.

[2] P. O'Hearn, J. Reynolds, and H. Yang. Local reasoning about programs that alter data structures. In *Computer Science Logic*, number 2142 in LNCS, pages 1–19, Paris, 2001.

[3] F. Smith, D. Walker, and G. Morrisett. Alias types. In *European Symposium on Programming*, pages 366–381, Berlin, Mar. 2000.