Destructive Logic

Runming Li
runmingl@andrew.cmu.edu
Carnegie Mellon University

March 19, 2022

Abstract

Intuitionistic logic, also know as constructive logic, is considered the true logic of computer science for the reason that it contains actual computational content. We propose Destructive logic, a supplement of constructive logic, with the goal of proving things that are not provable in constructive logic. Destructive logic, suggested by its very name, proves things by destroying, rather than construction. We study its properties and applications in this paper.

0 Introduction[0]

Constructive logic was historically proposed in response to classical logic. Everything that is provable in constructive logic is provable in classical logic, but not vice versa (e.g. law of excluded middle). In this sense, constructive logic is a more restricted, more precise logic than classical logic. It was later found out that constructive logic has roots in type theory and programming language theory based on Curry-Howard Isomorphism[1]. Constructive logic is so well-studied that many major universities teach it. Among them one of the classical[2] courses is 15-317 Constructive Logic at Carnegie Mellon University.

Duality exists everywhere in logic: conjunction and disjunction, universal and existential, left and right, up and down, you and me. A natural question to ask is: what is the dual of constructive logic? An unsurprising answer from etymology suggests destructive logic. Indeed this is the case: while constructive logic proves things by providing a clear algorithm that constructs the object in question, destructive logic proves things by destroying things it wants to prove. Of course, unreasonable destruction is considered psychopath, so we need to develop a clear justification for every destruction. For the purpose of destructive logic, introduce  , named *Destro* (they/them/theirs), a rational T-rex whose main purpose of life is to destroy things they do not like.

[0] It is well-known that a good logician starts counting from 0.

[1] Also know as *Propositions as types* principle. Also know as *Programs as proofs* principle. Also know as *Brouwer-Heyting-Kolmogorov interpretation*. Also know as *Realizability interpretation*. Also know as *Curry-Howard Correspondence*. It is worth noting that Curry-Howard Isomorphism is in no sense an isomorphism. Just like hotdog is in no sense a dog.

[2] Classical course, in a constructive sense.

1 Extending Natural Deduction

Gentzen[2] proposed natural deduction system to formulate intuitionistic logic. We extend his natural deduction system in support of our destructive logic.

1.1 Conjunction

In constructive logic, conjunction can be summarized by the following rules.

$$\frac{\Gamma \vdash A \ true \quad \Gamma \vdash B \ true}{\Gamma \vdash A \wedge B \ true} \wedge I \qquad \frac{\Gamma \vdash A \wedge B \ true}{\Gamma \vdash A \ true} \wedge E_1 \qquad \frac{\Gamma \vdash A \wedge B \ true}{\Gamma \vdash B \ true} \wedge E_2$$

Destro feels uneasy about this kind of setup: everything is too deterministic, and they want to attack propositions they do not like. Admittedly Destro's personal taste is too hard to predict, so we have the following possibilities.

$$\frac{\Gamma \vdash A \ true \quad \Gamma \vdash B \ true}{\Gamma \vdash \text{🦖} \wedge B \ true} \wedge I_1 \qquad \frac{\Gamma \vdash A \ true \quad \Gamma \vdash B \ true}{\Gamma \vdash A \wedge \text{🦖} \ true} \wedge I_2$$

$$\frac{\Gamma \vdash \text{🦖} \ true \quad \Gamma \vdash B \ true}{\Gamma \vdash A \wedge B \ true} \wedge I_3 \qquad \frac{\Gamma \vdash A \ true \quad \Gamma \vdash \text{🦖} \ true}{\Gamma \vdash A \wedge B \ true} \wedge I_4$$

$$\frac{\Gamma \vdash A \ true \quad \Gamma \vdash B \ true}{\Gamma \vdash \text{🦖} \wedge \text{🦖} \ true} \wedge I_5 \qquad \frac{\Gamma \vdash \text{🦖} \ true \quad \Gamma \vdash B \ true}{\Gamma \vdash \text{🦖} \wedge B \ true} \wedge I_6$$

Again, Destro's destruction is non-deterministic (at least humans should never be able to learn a T-rex's personal opinion), so we would not bother writing out all the possibilities in which they can attack for the purpose of saving some space. Moreover, one day Destro is tired of destructing propositions, so they start to destroy rules and context and even turnstile.

$$\frac{\text{🦖} \vdash \text{🦖} \ true \quad \Gamma\text{🦖}\text{🦖} \ true}{\Gamma \vdash \text{🦖} \wedge \text{🦖} \ true} \wedge \text{🦖}$$

1.2 Disjunction, Implication, Truth, and Falsity

Now one can simply img🦖ine what hap🦖en when we decide to put 🦖into other con🦖ectives such as disjunc🦖ion, implic🦖tion, truth, and falsi🦖y. In short, they just destroy whatever they want.

2 Properties and Applications

2.1 Basic Theorems

We first prove some basic theorems of destructive logic to convince the readers that indeed destructive logic is a real logic.

Theorem 1. *Consistency: it is not the case that $\bot$ true.*

Proof. In Curry-Howard Isomorphism, $\bot$ corresponds to **0** the empty type. 🦖cannot resist destroying it because it looks so much similar to their baby egg. Therefore, whenever the prover is able to show $\bot$ *true*, 🦖destroys it. □

Theorem 2. *Local completeness of conjunction: the elimination rules of conjunction are not too weak.*

Proof. Imagine a day when 🦖is too tired and they decide not to destroy anything. Then for that day destructive logic downgrades to constructive logic, in which local completeness of conjunction holds. □

Corollary 2.1. *Local completeness of other connectives.*

Proof. Copy and paste the proof of Theorem 2. □

Theorem 3. *Local soundne🦖s of conjunction: the elimination rules of conjunction are not too strong.*

Proof. Local soundne🦖s has been destroyed by 🦖. □

Theorem 4. *Global completeness and global soundness in destructive logic.*

Proof. The author has been destroyed by 🦖, and therefore leave the proof of this theorem to readers. □

2.2 Curry-Howard Isomorphism

Behind every logic there is a type system, according to Curry-Howard Isomorphism: constructive logic has simply typed lambda calculus, classical logic has continuation, linear logic has linear type system, second order logic has system F. It is only natural that destructive logic also has a corresponding type system. For this purpose, we propose υ-calculus[3]. We extend simply typed λ-calculus with the following types and terms.

$$\tau ::= \cdots \mid 🦖 \mid \tau^{\tau} \mid \texttt{unknown\_destroy}$$

$$e ::= \cdots \mid 🦖 \mid \upsilon(e, e) \mid \texttt{destro}(e)$$

The statics would be as followed.

$$\frac{}{\Gamma \vdash 🦖 : 🦖} \qquad \frac{}{\Gamma 🦖 e : \texttt{unknown\_destroy}} \qquad \frac{}{🦖 \vdash e : \texttt{unknown\_destroy}}$$

$$\frac{}{🦖🦖 e : \texttt{unknown\_destroy}} \qquad \frac{}{🦖🦖🦖🦖🦖} \qquad \frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \upsilon(e_1, \texttt{destro}(e_2)) : \tau^{🦖}}$$

[3]The choice of greek letter υ here is not arbitrary, but rather forced. Type theorists have used up so many good greek letters: α is for equivalence, β is for reduction, γ is for substitutions, Π and Σ are for dependent types, δ and o are for Covid, to name a few.

In this calculus, when 🦖accidentally destroy something strange (such as context or turnstile), we simply give it a universal type of `unknown_destroy`. The essence is the υ expression, where if e_1 is the undestroyed expression and e_2 is its destroyed equivalence, then $\upsilon(e_1, \texttt{destro}(e_2))$ is the way to recover the computational meaning of what 🦖destroyed. Then it follows naturally that

$$\cdot \vdash \upsilon(\upsilon(🦖, \texttt{destro}(🦖)), \texttt{detsro}(\upsilon(🦖, \texttt{destro}(🦖)))) : 🦖^{🦖^{🦖}}$$

2.3 Application

We suggest that destructive logic and υ-calculus be used to analyze randomized algorithms, since 🦖's destruction pattern is essentially randomized.

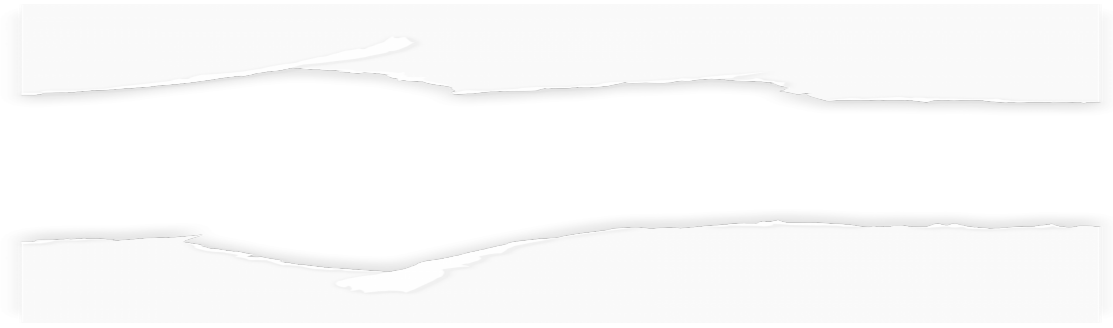
3 Future Work

We urge the logic community to take destructive logic seriously and put drastic amount of work in studying it. We further propose that destructive logic be taught at major universities. For Carnegie Mellon University, we propose to start a new course 15-407 Destructive Logic, and place it in the Principles of Programming Languages Concentration requirement. The choice of course number here is not arbitrary: according to Chuang and Wu[1], in order for a course to be considered a PL course, it has to have a Collatz number of 59. Indeed, 15407 has a Collatz number of 59 and should be considered as a PL course. Moreover, every course with a 7 in the course number is a good course (e.g. 15-317, 15-417, 98-317). It follows naturally that the course mascot should be 🦖, named *Destro*.

References

[1] Brandon Wu Gabriel Chuang. "What Lothar Collatz Thinks of the CMU Computer Science Curriculum". In: *SIGBOVIK 2021* (2021).

[2] Gerhard Gentzen. "Untersuchungen über das logische Schließen. II". In: *Mathematische Zeitschrift* 39.1 (1935), pp. 405–431. DOI: 10.1007/bf01201363.



This page has been destroyed by 🦖.