Logic of Programs	**Lecture 23**
CS 6860 Fall 2015	**Thurs. Nov 12, 2015**

Lecture 23

Relating Classical and Constructive Logics

We sketch a constructive semantics for classical logics that preserves the computational meaning of all the logical operators. We achieve this using the refinement type $\{x\!:\!A|B(x)\}$, a basic type of the Nuprl type theory since its creation, defined in [1]. This is the type of all elements a of A for which we possess constructive evidence b for $B(a)$. However, the evidence b is not retained with the type. We define classical logic using an instance of this type, simply writing $\{P\}$ for the type $\{Unit|P\}$. We also add one new axiom, proved to be consistent using the Coq formal model. We explain that next. Classical logic allows the *Law of Excluded Middle* (LEM), that any proposition P is either true or false, written symbolically as $P \vee \sim P$. This is *not* accepted in constructive logic where $P \vee \sim P$ means that we can find evidence for either P or for $\sim P$, and that evidence indicates the disjunct it validates. The constructive meaning of this logical law is that we have the computational ability to decide which case holds. On the other hand, the standard classical interpretation does not refer to human mental capabilities nor to computation. It assumes a mathematical reality where *truth* is independent of what we can discover about it, even with the aid of our mechanized proof assistants.

These different conceptions agree on the notion that $P \vee \sim P$ cannot possibly be (constructively) false. That is, we know $\sim\sim (P \vee \sim P)$. This is easy to understand because to know $\sim (P \vee \sim P)$ would be to know that there is no evidence for P. To know that is precisely to know $\sim P$. This common agreement is sufficient to allow a great deal of shared mathematics between classical and constructive proof assistants.

The logical truth expressed in the LEM, that for any proposition P either it or its negation, $\sim P$, is true can now be explained in terms of constructive evidence that does not refer to truth. *Virtual evidence* and the constructive impossibility of negative evidence are sufficient *semantic* grounds for knowing those propositions that are traditionally regarded as classically true. This semantics supports a clear and useful computational meaning as is explained in [**?**]. One new axiom creating *virtual evidence* retains the constructive meaning of all the logical operators. This semantics allows constructive proof assistants to use the results of classical proof assistants such as HOL [2] and PVS [6].

A constructive semantics seems impossible for classical logic because existence in constructive logic means that we can *construct the object* claimed to exist, whereas classical logic allows much weaker existence claims. To prove $\exists x{:}D.P(x)$ constructively requires that we show how to build an element d in the domain of discourse D, and then create evidence pd for $P(d)$. Classical logic judges that $\exists x{:}D.P(x)$ is true even when the only truth is that it is contradictory to assume that there is no object d such that $P(d)$; that is, the truth of

$\sim\sim \exists x{:}D.P(x)$ is sufficient to justify $\exists x{:}D.P(x)$. We will see that virtual existence can be expressed in the virtual evidence semantics, and it suffices for the classical existence claims.

We will show how to define *virtual constructive evidence* for *classical propositions* using the *refinement type* of computational type theory to specify the classical computational content. The refinement type, $\{Unit|P\}$, is critical. If P is known by constructive evidence p, then the refinement type has $\star$, the one and only element of $Unit$, as its *explicit* element; and p is hidden. We imagine that p has been "squashed" to $\star$, and we call $\{Unit|P\}$ "squashed P," or "classical P" and write it as $\{P\}$. Alternatively, we could take $\{P\}$ as a primitive classical logical operator with the above property. That approach is simpler if we are not exploring other aspects of constructive type theory.

The key new logical notion is expressed in the axiom $\sim\sim P \Rightarrow \{P\}$, where $\{P\}$ is the type $\{Unit|P\}$ where $Unit$ has a single element, $\star$. We call this *Classical Introduction* (CI). The justification is that we know constructively that $\{Unit|P\}$ cannot be empty. Since it can only have $\star$ as an element, the CI axiom allows that element as a member. This axiom creates *virtual evidence* for P even when there is no actual evidence for it, and we know that there is no evidence of any kind for $\sim P$. This axiom is not justified by explanatory constructive evidence, but we know that the new axiom does not allow us to prove $False$. So CI is constructively known to be *not false*. This mixed mode expression has a trivial realizer, the constant function with value $\star$. The virtual evidence can be used to *construct* virtual evidence in other refinement types by following the standard rules for refinement types and the other constructive types. The semantics for this axiom is another confirmation of Kolmogorov's insight that the law of double negation elimination (DNE), $\sim\sim P \Rightarrow P$, is the appropriate new axiom to create classical versions of a theory from the constructive ones. These ideas lead to the key insight that if we squash all types, making them classical, then a constructive logic becomes a classical logic.

The new axiom converts constructive evidence for $\sim\sim P$ into *virtual evidence* for $\{P\}$. We say that any evidence for P is *squashed* to the single unique element of $Unit$. We will see that this virtual evidence can be used in the constructive logical machinery. The axiom is similar to the law of *Double Negation Elimination* (DNE) favored by Kolmogorov. If there is constructive evidence for $\sim\sim P$, and none known for P, then there is only virtual evidence for $\{P\}$. That evidence is also hidden and can be un-hidden only when proving classical propositions. *Virtual constructive evidence carries more information than is provided by standard classical semantics.* This new semantics integrates classical and constructive logic and is possible because constructive type theory expresses considerably more kinds of evidence than classical set theory, the standard semantic basis for classical logic in mathematics. The treatment of existence follows Gödel's method of syntactically embedding classical logic into constructive logic. We give his method here. It requires only that $\sim \forall x{:}D. \sim P(x)$ is constructively true to justify classical $\exists x{:}D.P(x)$.

The young A. Kolmogorov showed in 1924/25 how to translate classical propositional logic into constructive propositional logic using $\sim\sim P$ in place of P. His method was extended by Kuroda [5] to first-order logic and modified by Gödel for number theory (see [3]). That method and its modifications do not provide a new *semantic account* of classical logic, rather they rely on *syntactically translating* classical propositions to constructive ones, and classical proofs to constructive ones. Ideas closely related to the approach presented here were studied by A. Kopylov and A. Nogin [4] using a classical metatheory and Russian constructive mathematics to establish *Markov's Principle* for propositional type theory. Their ideas are discussed further in the article. One might claim that our virtual semantics is constructive in the Russian sense. It definitely justifies Markov's principle. When the Classical Introduction axiom is integrated into a constructive logic, it allows us to prove all of the standard results of classical logic as well as propositions that mix the classical and constructive modes of reasoning. It holds for theories as rich as constructive type theory itself. We have added the law to the constructive type theory implemented by the Nuprl proof assistant, CTT. In the formal semantics for CTT, we can prove that this new axiom is not contradictory. Propositions proved without this axiom have *concrete evidence* and *computational content.* All propositions proved using it have *virtual evidence.*

References

[1] Robert L. Constable. Constructive mathematics as a programming logic I: Some principles of theory. In *Annals of Mathematics*, volume 24, pages 21–37. Elsevier Science Publishers, B.V. (North-Holland), 1985. Reprinted from *Topics in the Theory of Computation*, Selected Papers of the International Conference on Foundations of Computation Theory, FCT '83.

[2] Michael Gordon and Tom Melham. *Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic.* Cambridge University Press, Cambridge, 1993.

[3] S. C. Kleene. *Introduction to Metamathematics.* D. Van Nostrand, Princeton, 1952.

[4] Alexei Kopylov and Aleksey Nogin. Markov's principle for propositional type theory. In L. Fribourg, editor, *Computer Science Logic, Proceedings of the 10th Annual Conference of the EACSL*, volume 2142 of *Lecture Notes in Computer Science*, pages 570–584. Springer-Verlag, 2001.

[5] S. Kuroda. Intuitionistische Untersuchungen der formalistchen Logik. *Nagoya Mathematical Journal*, 2:35–47, 1951.

[6] S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, June 1992. Springer-Verlag.