Explicit logic is a great tradition similar to that of classical and intuitionistic logic. Due to its very explicitness, it is the most challenging of the three, and the one most closely connected with Computer Science. Languages whose most natural semantics are in explicit logic include functional programming languages, PROLOG, together with theorem provers such as Nuprl, HOL, Coq, etc.

From the point of view of logic, every computation is a deduction from initial premises in a suitable associated formal system. From the point of view of Computer Science, every deduction is an execution sequence of a computation with the premises as initial data. The "Greats" (Brouwer, Kolmogorov, Goedel) raised in the 1930's a complex of fundamental questions of explicit logic which we can now characterize as determining the exact relation between deduction and computation. One such fundamental relation, stemming from the consistency proofs of Gentzen, has already become a cornerstone of Computer Science. This is the Curry-Howard Isomorphism between typed lambda-terms and intuitionistic deductions.

The Curry-Howard Isomorphism does not of itself capture the self-referential mechanisms available in deductions in sufficiently rich formal systems and in programming languages. Our discovery of a natural system of self-referential proof terms, which we call "proof polynomials", was essential in our solution to the famous open problem of Goedel (1933) concerning formalization of provability (cf. major scientific accomplishments). Proof polynomials considerably extend the Curry-Howard Isomorphism and lead to a joint calculus of propositions and proofs which unifies several previously unrelated areas. We will explain our solution, and why it changes our conception of the appropriate syntax and semantics for typed programming languages, automated deduction and formal verification, reasoning about knowledge, nonmonotonic reasoning.

Work in progress:

- Reflective lambda-calculus which matches the power of proof polynomials. So far, we have succeeded in building the reflective lambda-calculus (confluent and strongly normalizable) for the core underlying system based on the intuitionistic propositions with implication and conjunction. Next steps here: to capture universal quantifiers, disjunction, classical logic, second order abstraction.
- Building feasible reflection in Nuprl and related systems. An ability of a language to talk about its own objects (sentences, types, proofs) is an essential component of its expressive power. In order to catch up with the conciseness of human reasoning a formal deductive system should have a tractable reflection mechanism. Traditional Goedel numbering is notoriously inefficient and cannot be implemented without principal reconstruction which constitute a challenging theoretical and practical problem.
- Explicit extensibility mechanism for verification systems. A typical problem of verification is to establish a fact \( F \) by means of a given formal system \( V \) by verifying in \( V \) that a certain construction \( D \) is a proof of \( F \). Such a scheme necessarily needs a reflection rule of the sort: "\( D \) is a proof of \( F \)" yields \( F \). The traditional (implicit) reflection does not
The formal verification with the implicit reflection rule is impossible since the justification of such a rule takes the "reflection tower" of theories of increasing proof-theoretical strength well beyond $V$. In turn, the explicit reflection is provable inside the system, which allows circumventing the "reflection tower" and provides the desired justification of verification.

- Logics of knowledge with justifications, where the atoms "$F$ is known" are replaced by the more informative "$t$ is an evidence of $F$". The desire to have such explicit epistemic systems has been repeatedly mentioned by the experts in the corresponding fields of Artificial Intelligence and knowledge representation.
- Explicit provability systems for the higher order theories which presumably provide the provability semantics for the polymorphic second order lambda-calculus. Explicit provability semantics for Girard's system $\text{F}$ and the Martin-Lof type theory $\text{ITT}$.
- Explicit non-monotonic reasoning on the basis of the explicit modal logic; the goal here is to find a concise non-monotonic logic that subsumes all major non-monotonic systems.

Applications:

In particular, I am working within the Cornell University PRL-team (R. Constable) on the reflection mechanism design for the next generation Nuprl style automated problem-solving systems (Nurpl 5 and MetaPrl). The current system is Nuprl 4, a 200,000-line Lisp-ML hybrid. It is being used for circuit synthesis; program language semantics; and formal verification; as accumulated database of mathematical knowledge. The system now ranks as one of the major tactic-oriented theorem provers.