Below is a summary of the contributions made by my research group over the past 20+ years. Only the big ideas get a place here. My group’s research has been centered around computational logic, logic programming, and its applications, as well as assistive technology. Along the way, my group has solved several problems that were considered “unsolvable.”

1. **Analysis of Or-parallelism:** The major result achieved here was to show that or-parallel search (as in logic programming or AI) cannot be parallelized without incurring a non-constant time overhead: i.e., it is not possible to devise a scheme that will perform all operations involved in an (or-parallel) search in constant time. The result first appeared in ACM TOPLAS; a more formal proof appeared in New Generation Computing. This result also allowed various models for realizing or-parallelism to be categorized and classified. Big step forward in understanding or-parallelism (recall that the Japanese Fifth Generation project all but abandoned or-parallelism).


2. **Incremental Stack-Splitting:** A scalable technique for implementing or-parallelism:
   This technique was a culmination of the extensive work that my group did in the 90s in realizing and-or parallel execution models for logic programs and studying the problem of supporting multiple environments in or-parallel execution. Stack-splitting is an extension of the stack-copying technique devised by Khayri Ali of SICS. The stack-splitting technique is scalable in that it can be used to realize or-parallelism on a a large number of processors that may or may not share memory. It has been deployed for building all types of or-parallel systems, particularly by Santos Costa and Rocha’s group in Porto. I believe it to be the best technique for parallelizing search.


3. **Tabled Logic based on Dynamic Reordering of Alternatives:** The DRA techniques realizes tabling in logic programming by repeatedly reordering the alternatives in Prolog’s search tree at runtime. The main advantage is the simplicity of this technique as well as its space-efficiency. The method influenced Neng-Fa Zhou’s linear tabling method included in the B-Prolog system. Our work on DRA also lead to invention of mode-directed tabling that allows dynamic programming problems to be elegantly solved within logic programming. Neng-Fa Zhou has incorporated mode-directed tabling in his PICAT system and used it for solving planning problems. Mode-directed tabled LP based planning in PICAT competes with the best available planners. The mode-directed tabling paper received the “most practical paper” award at PADL2004.


4. Horn Logical Denotational Semantics: The idea is to use logic programming to express denotational semantics. Both syntax and semantics can be specified as logic programs, and an interpreter obtained (so we have both “executable syntax” and “executable semantics”). Compiled code can be produced via partial evaluation (first Futamura projection). There are many applications: from provably correct code generation, to rapidly implementing domain specific languages, to processing languages that only admit context sensitive grammars, to rapidly building provably correct translators. The idea generalizes to continuation semantics quite elegantly. Horn logic denotations is the technology behind Interoperate, Inc., one of the two companies I founded.


5. Faithfully Modeling Real-time Systems with CLP(R): It is hard to model real-time systems faithfully (as time is continuous). Most systems model time by discretizing it. Discretizing time only produces an approximation. This work showed how time in real-time systems can be faithfully modeled using constraint logic programming over reals (CLP(R)). We began by showing how timed automata can be faithfully modeled; this was then generalized to timed grammars, timed push down automata, timed pi-calculus, timed linear temporal logic, timed planning, etc.


6. Nemeth Braille Math Code to Print PDF Translation: This problem involved automatic translation of mathematics and text written using Nemeth code and contracted Braille, respectively, to print PDF so that a sighted person can read what a blind person has brailled. This problem is called the backtranslation problem. It is useful in the classroom where the pupil is blind and the teacher sighted. Nemeth code was developed (1951) before formal study of syntax was started by Chomsky (1957). Nemeth code is a context sensitive language and is specified via examples. The backtranslation problem was widely considered unsolvable. My group solved it using logic programming and Horn clause semantics approach. The research also resulted in a company that won many SBIR awards, and produced the BrailleMath product.


7. An Aural Language for Voice-based Browsing: The idea is to have a formal language that can be spoken to give commands to an aural browser. This system gives the user the ability to mark specific points in the passage being read by the aural browser through voice utterances.
Complex navigation strategies can then be created and aurally spoken by the user based on these voice-marks.


8. **Constraint Spreadsheets**: The idea is to generalize a spreadsheet to support finite domain constraints. It has been implemented via at least 5 student MS theses, the latest one being the PlanEx tool in Abhilash Tiwari’s MS thesis in 2009. An earlier version of the system was successfully used to automatically design CS and EE Department’s course schedule.


9. **Coinductive Logic Programming**: The idea is to give operational semantics to compute answers that are in the greatest fixpoint semantics. Coinductive LP has far-reaching applications. Standard logic programming (or standard computation) is based on induction (least fixpoint semantics). Infinite structures such as perpetual programs, ω-automata, etc., cannot be modeled (at least elegantly) by induction. Rather, they need coinduction for modeling. The paper that introduced coinductive LP received the ICLP 2016 test-of-time award for being the most influential paper of ICLP 2006.


10. **Query-driven Answer Set Programming**: This problem pertains to executing answer set programs in a goal-directed, top-down manner. This problem was widely considered unsolvable and most implementations are based on using a SAT solver, and thus can only handle propositional programs. The idea of coinductive logic programming is what facilitated the first attack on this problem. It culminated in the Galliwasp system, the first of its kind. Galliwasp was then generalized to the s(ASP) system that can execute predicate answer set programs directly. All other systems can only handle propositional answer set programs. Both Galliwasp and s(ASP) are publicly available. s(ASP) has been used to develop a number of innovative applications. A student hackathon was recently organized around it.

