Summary of Research Contributions
Gopal Gupta
10th August, 2017

Below is a summary of the contributions made by my research group over the past 20+ years (in reverse chronological order). Only major ideas are summarized. My research has been centered around computational logic, logic programming, and its applications, as well as assistive technology. My group has solved several problems that were considered impossible to solve by other researchers.

1. **Inductive Learning of Default Theories**: This line of research extends work done in the field of inductive logic programming (where one learns a logic program from examples) to learning normal logic programs, i.e., learning logic programs that contain negation as failure. Normal logic programs have been shown to model human-style common sense reasoning quite well. The inductive learning algorithms my group has developed learns a default relation as well as exceptions to it. The learning algorithm can also handle features that range over real numbers. The algorithms recursively learn exceptions to exceptions and so on thereby producing a learned theory that has high accuracy. The effort is a step towards what has been dubbed *Explainable AI*. A prototype implementation has been developed.

   ◇ Farhad Shakerin, Elmer Salazar, and Gopal Gupta. Inductive Learning of Default Theories. *Proc. International Conference on Logic Programming (ICLP)*, Journal of Theory and Practice of Logic Programming, Cambridge University Press, to appear.

   .

2. **Query-driven (Predicate) 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 pioneered by my group is what facilitated the first stab at 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 (where predicates can contain arbitrary terms). All other systems available thus far 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.

   ◇ Kyle Marple, Ajay Bansal, Richard Min and Gopal Gupta. Goal-Directed Execution of Answer Set Programs. *Proc. Principles and Practice of Declarative Programming (PPDP)*, ACM Press, 2012. pp 35-44.
   ◇ Kyle Marple and Gopal Gupta. Galliwasp: A Goal-Directed Answer Set Solver. Selected Papers from LOPSTR'12. Springer LNCS 7844. pp. 122-136. 2013.
   ◇ Kyle Marple, Elmer Salazar, Zhuo Chen, Gopal Gupta. The s(ASP) Predicate Answer Set Programming System. ALP Newsletter. March 2017.

3. **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. Coinductive logic programming has been in many Prolog implementations, most notably in SWI Prolog.

- ◇ L. Simon, A. Mallya, A. Bansal, G. Gupta. Coinductive Logic Programming. In *Proc. Int'l Conference on Logic Programming*. 2006. Springer Verlag LNCS 4079. pp. 330-345.
- ◇ G. Gupta, N. Saeedloei, R. Min, B. DeVries, K. Marple, F. Kluzniak. Infinite Computation, Co-induction, and Computational Logic. In *Proc. 4th International Conference on Algebra and Co-algebra in Computer Science*. Springer Verlag, pp. 40-54.
- ◇ Neda Saeedloei, Gopal Gupta. Coinductive Constraint Logic Programming. FLOPS 2012. Springer LNCS 7294. pp. 243-259

4. **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.

   - ◇ G. Gupta, E. Pontelli. A Constraint-based Approach to Specification and Verification of Real-time Systems. In *Proc. IEEE Real-time System Symposium*, San Francisco, pp. 230-239. Dec. '97.
   - ◇ N. Saeedloei, G. Gupta. Timed Definite Clause ω-Grammars. Proc. 26th International Conference on Logic Programming. 2010. pp. 212-221
   - ◇ N. Saeedloei, G. Gupta. Timed π-calculus. Proc. 8th International Symposium on Trusted Global Computing. Buenos Aires. 2013. Springer Verlag, LNCS 7844. pp. 122-136.

5. **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.

   - ◇ A. Karshmer, G. Gupta, S. Geiger, C. Weaver. "A Framework for Translation of Braille Nemeth Math to Latex," In *Proc. ACM Conference on Assistive Technologies*, ACM Press, pp. 136-143, Mar. 1998.

6. **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.

   - ◇ M. Nichols, Q. Wang, G. Gupta. A VoiceXML-based Spoken Scripting Language for Voice-based Web Navigation. In *Human Computer Interaction* Conference, July 2005, Lawrence Erlbaum and Associates.

7. **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.

- ◇ G. Gupta, S. Akhter. Knowledgesheet: A Graphical Spreadsheet Interface for Interactively Developing A Class of Constraint Programs. In *Proc. Practical Aspects of Declarative Languages*, Lecture Notes in Computer Science 1753, Springer Verlag, 2000.

8. **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. Nearly all tabled Prolog systems incorporate our idea of mode-directed tabling, e.g., SWI, YAPTAB, XSB, PICAT. Mode-directed tabled LP based planning in PICAT, for instance, competes with the best available planners. The mode-directed tabling paper received the "most practical paper" award at PADL2004.

   - ◇ H-F. Guo, G. Gupta. A Simple Technique for Implementing Tabling based on Dynamic Reordering of Alternatives. *Proc. 17th Int'l Conf. on Logic Programming*, Papphos, Cyprus, Springer Verlag LNCS 2237. pp 181-198.
   - ◇ H-F. Guo, G. Gupta. Simplifying Dynamic Programming via Mode-directed Tabling. In *Proc. Sixth International Conference on Practical Aspects of Declarative Languages*. 2004. pp. 163-177.

9. **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.

   - ◇ G. Gupta Horn Logic Denotations and Their Applications. The Logic Programming Paradigm: A 25 year perspective. Springer Verlag. pp. 127-160. (Proceedings of Workshop on Current trends and Future Directions in Logic Programming Research, April '98).
   - ◇ Qian Wang, G. Gupta, M. Leuschel. Towards Provably Correct Code Generation via Horn Logical Continuation Semantics. in Proc. International Conf. on Practical Aspects of Declarative Languages 2005. Springer Verlag. LNCS 3350. pp. 98-112. 2005.

10. **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.

    - ◇ G. Gupta and E. Pontelli. Stack-splitting: A Simple Technique for Implementing Or-parallelism on Distributed Machines. In *Proc. 16th International Conference on Logic Programming*, 1999. MIT Press, pp. 290-305.

- ◇ E. Pontelli, K. Villaverde, H. Guo, G. Gupta. Stack Splitting: a Technique for Efficient Exploitation of Search Parallelism on Share-nothing Platforms. Journal of Parallel and Distributed Computing. 2006. pp. 1267-1293.

11. **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).

    - ◇ G. Gupta and B. Jayaraman "Analysis of Or-parallel Execution Models," *ACM Transactions On Programming Languages and Systems (ACM TOPLAS)*, Vol 15, No. 4, September 1993, pp. 659-680.
    - ◇ D. Ranjan, E. Pontelli, and G. Gupta. "On the Complexity of Or-parallelism," In *New Generation Computing: An International Journal* Vol. 17, No. 3, May 1999.