# Logic with Performance: The Hybrid Approach

Lyndon Henry
The University of Sydney

October 10, 2019

**Abstract**

Logic programming is on the rise in both research and industry. Logic is being applied to real-world problems, bringing success where more traditional methods have failed. The problem is one of performance – logic engines lack speed and scalability. Our solution is the *hybrid approach*. We bring speed to logic programs matching that of problem-specific, hand-optimised imperative code. We bring scalability to logic programs competitive with existing distributed computing platforms. All of this is done fully automatically, with no change in the logic language from the perspective of the user.

In part one, we show how *extra-logic* relations can bring logic the speed of low-level implementations, while appearing indistinguishable to the user from native relations. We target points-to analysis, a form of static program analysis which has been a major application of logic programming. We map points-to analysis to a variant of directed graph reachability, then introduce a new method for this problem. Significantly, the new method performs better than current state of the art graph reachability algorithms across all datasets tested, both in runtime and memory. The new method is integrated into a logic engine via extra-logic relations, bringing efficient algorithms and data-structures that would not be possible to implement within a logic language.

In part two, we introduce a new distributed bottom-up logic evaluation, transforming logic programs into computations on *streams*, thus bringing logic scalability. Logic programs are broken up into subprograms or *actors*, with dependencies between subprograms becoming communication channels. Actors continually stream information, interleaving computation and communication, while working in parallel across both the nodes of the network, and the cores of each node. Our new approach achieves low latency and high utilisation, is non-locking, non-blocking, has no shared state, no coordination, and no synchronisation.