At the occasion of the Third International Conference on Principles of Knowledge Representation and Reasoning [1] in 1992, Ray Reiter delivered an invited talk entitled “Twelve Years of Nonmonotonic Reasoning Research: Where (and What) Is the beef?” [2,3] reflecting the state and future of the research area of Nonmonotonic Reasoning (NMR;[2]). Ray Reiter describes it in [3] as a “flourishing subculture” making many outside researchers “wonder what on earth this stuff is good for.” Although he seemed to be rather optimistic about the future of NMR, he nonetheless saw its major contribution on the theoretical side, providing “important insights about, and solutions to, many outstanding problems, not only in AI but in computer science in general.” Among them, he lists “Logic Programming implementations of nonmonotonic reasoning”.

Although the link between Michael Gelfond and Vladimir Lifschitz’ Stable Model Semantics for Logic Programming [4] and NMR formalisms like Ray Reiter’s Default Logic [5] were discovered soon after the proposal of Stable Model Semantics, it still took some years until the first such implementation was conceived, namely, the *smodels* system [8,9]. The emergence of such a highly efficient and robust system has boosted the combination of Logic Programming and NMR and finally led to a novel declarative programming paradigm, referred to as Answer Set Programming (ASP;[10,11,12,13,14]). Since its inception, ASP has been regarded as the computational embodiment of Nonmonotonic Reasoning and a primary candidate for an effective tool for Knowledge Representation and Reasoning. After all, it seems nowadays hard to dispute that ASP brought new life to Logic Programming and NMR research and has become a major driving force for these two fields, helping dispel gloomy prophecies of their impending demise.

Meanwhile, the prospect of ASP has been demonstrated in numerous application scenarios, including bio-informatics [15,16], configuration [17], database integration [18], diagnosis [19], hardware design [20], insurance industry [21], model checking [22], phylogenesis [23,24], planing [12], security protocols [25], etc. A highlight among these applications is arguably the usage of ASP for the high-level control of the space shuttle [26,27]. The increasing popularity of ASP is for one thing due to the

---

1. By then twelve years after the publication of the Special Issue of the Artificial Intelligence Journal on Nonmonotonic Reasoning.
2. See also [http://en.wikipedia.org/wiki/Where’s_the_beef](http://en.wikipedia.org/wiki/Where’s_the_beef)
3. Logic Programming under Stable Model Semantics turned out to be a special case of Default Logic, with stable models corresponding to default extensions [6,7].
4. See also [http://www.kr.tuwien.ac.at/research/projects/WASP/report.html](http://www.kr.tuwien.ac.at/research/projects/WASP/report.html)
availability of efficient off-the-shelf ASP systems \cite{28,29,30,31,32} and for another due to its rich modeling language, jointly allowing for an easy yet efficient handling of knowledge-intensive applications. Essentially all ASP systems that have been developed so far contain two major components. The first of them, a **grounder**, grounds an input program, that is, produces its compact propositional equivalent, often by appeal to advanced database techniques. The input language goes well beyond that of Prolog, offering among others, integrity constraints, classical negation, disjunction, and various types of aggregates. The second component, a **solver**, accepts the ground program and actually computes its answer sets (which amount to the stable models of the original program). Modern ASP solvers rely on advanced Boolean constraint solving techniques, stemming from the area of Satisfiability Checking and allowing for tackling application problems encompassing millions of variables. All in all, ASP has become an efficient and expressive declarative problem solving paradigm, particularly well-suited for knowledge-intensive applications.

Taking up Ray Reiter’s challenge after sixteen years, *my* obvious answer is that Answer Set Programming is the *beef* of twenty-eight years of NMR research! Although twenty-eight years appear to be quite a while, successful neighboring areas such as Description Logics (DLs) and Satisfiability Checking (SAT) look back onto similar histories, taking major references in their field, like \cite{33} and \cite{34,35}, respectively. Nonetheless both areas have prospered in recent years due to their success in industrially relevant application areas. SAT is the key technology underlying Bounded Model Checking \cite{36} and DLs have become standard ontology languages for the Semantic Web \cite{37}. Although different factors have abetted these success stories, in the end, their breakthrough was marked by their establishment as salient technologies in their respective application areas. What can ASP learn from this? First of all, we should keep building upon strong formal foundations, just as SAT and DLs do. However, ASP should gear its research vision towards application scenarios in order to make ASP technology more efficient, more robust, more versatile, and in the end ready for real applications. This orientation is such a fruitful approach, being full of interesting and often fundamental research questions.

Second, we have to foster the dissemination of ASP in order to increase its perception. Apart from promoting ASP in our academic and industrial environment, teaching ASP is an important means to enhance the common awareness of it. This does not necessarily mean to teach full-fledged ASP courses, which is difficult in view of many encrusted curricula, but rather to incorporate ASP in AI-related classes as a tool for illustrating typical reasoning patterns in Knowledge Representation and Reasoning, like closed-world reasoning, abduction, planning, etc. And after all, to put it in Ray Reiter’s words, it’s the ASP community’s duty to show “what on earth this stuff is good for.”

ASP has staked its claim in being an attractive approach to declarative problem solving in combing an expressive modelling language with efficient solving technology. But how does it scale? In fact, this is not only a matter of performance but also of applicability and usability. Here is my personal view.

**Performance.** Modern ASP solvers are based on advanced Boolean constraint technology and exhibit a similar performance as advanced SAT solvers \cite{38}. Unlike SAT, however, ASP offers a uniform modelling language admitting variables. In