Personal Perspective on the Development of Logic Programming Based KR Languages

Michael Gelfond

Texas Tech University

August 8, 2010

The Starting Point

In the early 80s I became interested in declarative programming, knowledge representation, and non-monotonic logics.

My early attempts to understand non-monotonicity lead to establishing a close relationship between Autoepistemic Logic and the programming language Prolog, and to the discovery of Stable Model Semantics of logic programs with negation as failure.

Gelfond, *On Stratified Autoepistemic Theories* (1987)

Gelfond and Lifschitz, *The Stable Model Semantics for Normal Logic Programs* (1988)

Normal Logic Programs with Stable Model Semantics

A *normal logic program* Π consists of a signature σ and a collection of *rules* of the form:

$$a_0 \leftarrow a_1, \ldots, a_m, \text{ not } a_{m+1}, \ldots, \text{ not } a_n$$

where a's are atoms of σ.

Π defines a collection of *stable models* — sets of beliefs to be held by a rational reasoner associated with Π.

Stable Model Semantics (continued)

Informally, stable models are collections of ground atoms which:

- Satisfy the rules of Π.
- Adhere to the *the rationality principle* which says: *Believe nothing you are not forced to believe.*

Program

$$a \leftarrow \text{not } b$$

has one stable model, $\{a\}$. The agent associated with the program believes a to be true and b to be false.

Program

$$a \leftarrow \text{not } a$$

has no stable model.

Mathematical definition of stable models captures this intuition.

Well-Founded Semantics

At about the same time, Well-Founded Semantics was introduced by Van Gelder, Ross, and Schlipf.

Under this semantics any normal logic program has a unique, three-valued intended model, e.g.

$$a \leftarrow \ not\ a$$

has the model in which a is undefined.

Discussion: Which is the Right Semantics?

Immediately after this introduction, discussion began about the relative merits of both semantics from the KR stand point.

Differences of opinion were caused not only by personal intuitions and tastes, but also by different views on

- the purpose of KR languages, and
- the proper methodology for KR research.

These differences, however, were often hidden in the background and very rarely articulated.

The Purpose of this Talk

Since this and other similar discussions continue today, I thought that it may be useful to talk about my views on the goals and the methodology of KR-language research.

We do not necessarily need to reach consensus on the subject, but I believe that asking and answering these questions is essential for every researcher.

We need to discuss them more often outside of the anonymous reviewing process.

I'll start with recalling the discussion which started around 1988, and my reaction to it.

Proposed Criteria for Language Evaluation

(1) Connectives of a formal language should have a reasonably clear intuitive meaning.

(2) The corresponding mathematics should be simple and elegant.

(3) The language should suggest systematic and elaboration tolerant representations of a broad class of phenomena of natural language. This includes basic categories such as belief, knowledge, causality, etc.

(4) A large number of interesting computational problems should be reducible to reasoning about theories formulated in this language.

Proposed Criteria for Language Evaluation (continued)

(5) Reasoning in the language should be efficient.

(6) Entailment relation, $\models$, of the language should satisfy some natural properties, e.g. if $T \models F$ and $T \models G$ then $T \cup \{F\} \models G$.

(7) Every program written in the language should be, in some sense, consistent.

(8) New language should extend the first-order classical logic.

The Purpose of KR Languages (Personal Perspective)

To decide which of the criteria are important I needed to better articulate why I want to represent knowledge. Here is a short answer:

- To better understand basic commonsense notions we use to think about the world: beliefs, knowledge, defaults, causality, intentions, probability, etc.
- To design and implement knowledge intensive software systems.

Applying the Criteria (Personal History)

I found the *first four criteria* crucial for a good language. The next four I discovered to be substantially less important and sometimes even harmful.

So I gave normal logic programs under Stable Model Semantics the following grades with respect to the criteria:

- B – clarity of intuition
- A – mathematical elegance
- C- – expressiveness
- I – adequacy for solving computational problems
- On criteria (5)–(8) the language failed.

Well-Founded Semantics had worse grades on (1) and (2) but fared much better on (5)–(8).

Improving Expressiveness: New Connectives

To improve expressiveness, the language of normal logic programs was extended to include two new connectives:

- *classical (strong, explicit) negation*

"$\neg a$" — a is false.
(as opposed to "not a" — belief in the truth of a is not justified.)

- *epistemic disjunction.*

"a or b" — the reasoner must believe a or must believe b.
(Hence "a or $\neg a$" is not a tautology.)

Answer Set Prolog (Gelfond and Lifschitz)

A *program* of ASP consists of a signature σ and a collection of *rules* of the form:

$$l_0 \text{ or } \ldots \text{ or } l_i \leftarrow l_{i+1}, \ldots, l_m, \text{ not } l_{m+1}, \ldots, \text{ not } l_n$$

where l's are literals of σ, (i.e. $l = p$ or $l = \neg p$).

The intuition and mathematical definition is very close to that of stable models.

But a set of beliefs (called an answer set) is now a set of literals!

Answer Set Prolog (continued)

Program

$$a \leftarrow \ not\ b$$

has the answer set $\{a\}$.

The agent associated with the program believes that a is true but is undecided about the truth value of b. (No closed world assumption!)

The answer set of the program

$$a \leftarrow \ not\ b \qquad\qquad \neg b \leftarrow \ not\ b$$

is $\{a, \neg b\}$. The agent believes that a is true and b is false.

Methodology

So what research methodology to use to evaluate our language?

• Initial Modeling of Basic Concept:

Emphasis on faithfulness to the intuition and mathematical accuracy and elegance of the model.

• Evaluation of the model by its use in designing small experimental systems capable of performing intelligent tasks.

The emphasis here is on the ability of the model to guide our design, generalizability of solutions, and discovery of new phenomena or a new perspective on the old one.

• Evaluation of the language by its use in design and implementation of midsize practical intelligent software systems.

Emphasis on efficiency, correctness, and degree of elaboration tolerance.

What has been Achieved? (Personal Perspective)

- It became clear rather early that

ASP preserves the degree of clarity of intuition and mathematical elegance of the original language.

- But much time and effort was required to show that:

(1) ASP is much more expressive than the language of normal logic programs.

(2) ASP is adequate for solving a large number of interesting computational problems.

- No improvement has been made in criteria (5) – (8).

Expressiveness

Ways were found to use ASP to represent:

- Rational Beliefs.
- Defaults and their exceptions.
- Causal effects of actions.

I believe that these results can serve as examples of substantial advances in KR. The next few slides elaborate on this.

Defaults

The default

"*normally elements of class* C *satisfy property* P"

can be expressed as

$$\begin{aligned} p(X) \leftarrow\ & c(X), \\ & \text{not } ab(d(X)), \\ & \text{not } \neg p(X). \end{aligned}$$

where $d(X)$ is the default's name.

Exceptions to Defaults

$c_1 \subset c$ is a *strong exception* to default $d(X)$:

$$\neg p(X) \leftarrow c_1(X).$$

$$ab(d(X)) \leftarrow \ not \ \neg c_1(X).$$

$c_2 \subset c$ is a *weak exception* to default $d(X)$:

$$ab(d(X)) \leftarrow \ not \ \neg c_2(X).$$

If information about membership in class C is complete, the representation can be simplified.

Defaults and Reasoning by Cases

Program

$$
\begin{array}{lcl}
c_1(a) & \text{or} & c_2(a). \\
p_1(X) & \leftarrow & c_1(X), \\
 & & \text{not } \neg p_1(X). \\
p_2(X) & \leftarrow & c_2(X), \\
 & & \text{not } \neg p_2(X). \\
q(X) & \leftarrow & p_1(X). \\
q(X) & \leftarrow & p_2(X).
\end{array}
$$

entails $q(a)$.

It is not clear how to model this type of reasoning in either LP under well-founded semantics or in Reiter's default logic.

Expressiveness: Effects of Actions

ASP is capable of elegantly expressing direct and indirect effects of actions, and addressing the frame and ramification problems.

For instance, the sentence

$$a \text{ causes } f$$

can be written as

$$holds(f, I+1) \leftarrow occurs(a, I).$$

The frame problem can be solved by the Inertia Axiom:

$$holds(F, I+1) \leftarrow holds(F, I),\ not\ \neg holds(F, I+1).$$

$$\neg holds(F, I+1) \leftarrow \neg holds(F, I),\ not\ holds(F, I+1).$$

Adequacy: Task Reductions

A large number of problems of high complexity, including planning and diagnostics, has been shown to be reducible to computing answer sets of logic programs. See, for instance,

- *Relating stable models and AI planning domains*, V.S. Subrahmanian and C. Zaniolo, 1995

- *Encoding planning problems in non-monotonic logic programs*, Dimopoulos, Koehler and Nebel, 1997

- *Diagnostic reasoning with A-Prolog*, M. Balduccini and M. Gelfond, 2002

Adequacy: Reasoning Systems

Development of

- efficient answer set solvers and
- Datalog and Prolog implementations

made possible a number of industrial applications built on these reductions.

ASP was shown to be useful for the design of industrial strength systems.

What is Next? (Personal History)

- The addition of explicit negation and epistemic disjunction improved the EXPRESSIVENESS of the language.
- ASP was shown to be ADEQUATE for solving a large number of computational problems.

DOES ASP NEED EXPANSION?

What we could not express in ASP

The answer seems to be "yes". We were not able to find ways to represent:

• Indirect exceptions to defaults: contradiction is found not with conclusion of the default but with consequences of this conclusion.

• Optimizations requirements: shortest plans, minimal diagnoses, etc.

• Weak constraints: e.g. "Perform action A only if it is impossible to get your goal without it".

CR-Prolog (Balduccini and Gelfond, 2003)

All this can be expressed in CR-Prolog, a language that expands ASP by *consistency-restoring* (CR) rules:

$$l \stackrel{+}{\leftarrow} body$$

and a partial order on the set of these rules.

A CR rule says that if the reasoner associated with the program believes the body of the rule then he *may, if necessary*, believe its head.

This can only be done if there is no way to obtain a consistent set of beliefs by using only regular rules of the program.

The partial order on sets of CR rules is used to select preferred possible resolutions of the conflict.

Indirect Exceptions in CR-Prolog

The *Contingency Axiom* (CA) for the default

$$\begin{aligned} p(X) \quad \leftarrow \quad & c(X), \\ & \text{not } ab(d(X)), \\ & \text{not } \neg p(X). \end{aligned}$$

has the form:

$$\neg p(X) \stackrel{+}{\leftarrow} c(X).$$

Let Π_1 consist of the two rules above and:

$$q(X) \leftarrow p(X). \qquad\qquad c(x).$$

Π_1 entails $q(x)$ without use of the CA. But

$$\Pi_2 = \Pi_1 \cup \neg q(x)$$

needs the CA to avoid contradiction and entail $\neg q(x)$.

P-log: Mixing Logical and Probabilistic Reasoning

P-log (Baral, Gelfond, Rushton) is an extension of ASP which allows combining of logical and probabilistic reasoning.

• P-log probabilities are defined with respect to an explicitly stated knowledge base.

• In addition to having logical non-monotonicity, P-log is "probabilistically non-monotonic" – new information can add new possible worlds and change the original probabilistic model.

• Possible updates include defaults, rules introducing new terms, observations, and deliberate actions in the sense of Pearl.

Conclusion

Much was accomplished by work on ASP.

- Foundations:

We better understand the intuition behind such basic notions as belief, defaults, probability, causality, intentions, etc.

This was done by the development of new mathematical theory and the methodology of its applications, and by experimental engineering.

This foundational work helped to put our science on solid ground.

Conclusion (continued)

- Building Systems:

We are learning how to use our theories to build transparent, elaboration tolerant, provably correct, and efficient systems.

Twenty years ago I didn't believe that such systems would be possible in my lifetime. I am obviously happy to be proven wrong.

For me, this work has been, and (I hope) will continue to be, deeply satisfying. I hope it will be equally satisfying for the new generations of researchers.