Answer Set Programming

A Story of
Default Negation, Definitions and Informal Semantics

Mirek Truszczynski
University of Kentucky
Lexington, KY, USA

A tribute to Ray Reiter

Answer Set Programming?

- A language for knowledge representation and reasoning
  - support for modeling commonsense reasoning phenomena
- A language for modeling and solving constraint satisfaction problems
  - a rich syntax for logical and numerical constraints
  - and for defining optimization criteria
- Declarative
- Well-understood expressive power
- And (perhaps most importantly) with *powerful computational tools*

Introducing Answer Set Programming

- To solve a search problem, a program is designed that captures the problem specifications so that when extended with facts that represent an instance of the problem, the answer sets of the resulting program describe all solutions of the problem for that instance

- The upshot of this design is that solving the problem is reduced in a uniform way (the program is fixed and only the data component changes) to the task of finding answer sets.

Brewka G., Eiter T., and Truszczynski M. *Answer Set Programming at a Glance.* Commun. ACM 54(12): 92-103 (2011)

Today I want to ...

- Look back at the key ideas and insights that lead to the formulation of the *ASP paradigm* and figured in its development

- Negation — default negation and negation as failure
- Inductive definitions, programs with negation as definitions
- Informal semantics of programs (and modeling methodology)
- Engineering fast automated reasoning tools
- Applications

Back to 1999 — the Year of the ASP Paradigm

- *Stable Models and an Alternative Logic Programming Paradigm*, Victor Marek & MT
- *Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm*, Ilkka Niemelä

Alternative LP Paradigm

- Consider a search problem Π (given by a set D_Π of *instances*, each instance $I \in D_\Pi$ coming with a set $S_\Pi(I)$ of solutions)
- Assume that there exist:
  - an effective encoding edb_Π under which every instance $I \in D_\Pi$ is represented as a set of ground atoms
  - a finite program, P_Π, such that for every instance I there is a computable one-to-one function sol^I_Π from the class of stable models of $edb_\Pi(I) \cup P_\Pi$ to $S_\Pi(I)$
- Then Π can be solved for an instance I by first constructing the program $edb_\Pi(I) \cup P_\Pi$, then by finding its stable model s and, finally, by reconstructing from s a solution $sol^I_\Pi(s)$.

Marek V.W., Truszczynski M. (1999) *Stable Models and an Alternative Logic Programming Paradigm*
In: Apt K.R., Marek V.W., Truszczynski M., Warren D.S. (eds) The Logic Programming Paradigm. Springer

Constraint Programming Paradigm

- We put forward logic programs with the stable model semantics (LP_{SM}) as an interesting constraint programming paradigm
- The underlying idea in this paradigm is to interpret the rules of a program as constraints on a solution set [of ground atoms]
- A logic programming system supporting the constraint interpretation of rules is very different from typical logic programming systems, such as Prolog implementations. Given a program the *main task of such a system is to compute solution sets, i.e., stable models, for the program.*

Niemelä, I. *Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm.* Ann. Math. Artif. Intell. 25(3-4): 241-273 (1999)

How Did We Get There?

- Automated theorem proving

- Logic for representing knowledge
- Logic for programming

John McCarthy

- Programs with *Common Sense* (McCarthy, 1958)
  - certain elementary verbal reasoning processes so simple that they can be carried out by any non-feeble minded human have yet to be simulated by machine programs
- Some philosophical problems from the standpoint of artificial intelligence (McCarthy and Hayes, 1969)
  - the first task is to define even a naive common-sense view of the world precisely enough to program a computer to act accordingly. This is a very difficult task in itself

Newell, 1980

- [] this basic engine is *not going to be powerful enough* to prove theorems that are hard on a human scale, [] or to serve other sorts of problem solving, such as robot planning
- *Uniform procedures will not work* — systems for AI must include basic mechanisms of problem solving (including logical inferences), but they also need ways to specify control
  (the view embodied by PLANNER and descendants by Hewitt and collaborators, Sussman, Winograd, and Charniak)
- Logical languages are themselves *not particularly good* formalisms for representing knowledge, nor is the application of rules of inference to logical formulas a particularly good method for *commonsense reasoning*

Moore, 1982

- There is an important set of issues, involving *incomplete knowledge* [] that, in some sense, probably can be dealt with only by systems based on logic and deduction
- The experiments of the late 1960s on problem-solving by theorem-proving did not show that the use of logic and deduction in AI systems was necessarily inefficient
- Deficiencies attributed to logical representations may be artifacts of *naive implementations* and do not necessarily carry over when more sophisticated techniques are used

Three Recurring Themes

- Commonsense reasoning
- Incomplete information
- Efficiency of reasoning

1980 AIJ Issue on Nonmonotonic Reasoning

- Circumscription by McCarthy
- Default logic by Reiter
- Modal nonmonotonic logics by McDermott and Doyle

Default Logic

- The need to make *default* assumptions is frequently encountered in reasoning about incompletely specified worlds
- Imagine a first order formalization of what it is we know about any reasonably complex world. Since we cannot know everything about that world [...] this first order theory will be *incomplete*
- Nevertheless, there will arise situations in which it is *necessary to act*. The role of a default is to *fill some of the gaps*, to further complete the underlying theory

Reiter R., *A Logic for Default Reasoning*
Artificial Intelligence 13, 81–137, 1980

Default Logic

- Defaults are then [...] *instructions* about how to create an extension of this incomplete theory
- Common reasoning patterns of inference that often take the form "*in the absence of information to the contrary, assume ...*"
- Reiter mentions here connections to CWA, THNOT in PLANNER by Hewitt, and to *not* in LP

Default Logic — Extensions

- While we could analogously [to McDermott and Doyle] define the theorems of a default theory as the intersection of its extensions, we choose not to pursue this point of view
- Instead our position is that the purpose of default reasoning is to determine *one* consistent set of beliefs about a world, i.e. *one extension*
- And to reason within this extension until such time as the evidence at hand forces a revision of those beliefs, in which case a *switch* to a new extension may be called for

A default formally

$$\frac{F : MG_1, \ldots, MG_k}{H}$$

- An *inference* rule
  - Premises of *two* types: a prerequisite F and justifications $G_1, \ldots, G_k$
  - Consequent H

Justifications — the Key to a Default

- Informal reading of a default

$$\frac{F : MG_1, \ldots, MG_k}{H}$$

  "if F and if it is *consistent to assume* each justification G_i, then infer H"

- How to read $M\neg G$?
  - $\neg G$ can be consistently assumed
  - $\neg G$ is *possible* — hence, M
  - there is *no* reason to hold G

- *Another* type of negation?

A Default Theory

- A pair (D, W)
  - D is a set of defaults (rules to close gaps in our knowledge)
  - W is a set of formulas we take we know

Its Meaning?

- A constraint on sets of possible worlds
  - in other words, on S5-models
- Each specified by a theory Reiter referred to as an *extension*

How used?

- A reasoner commits to an extension (to the set of possible worlds)
- Then uses this extension to make inferences and answer queries (with tools for FO provability)
- When the default theory changes, all that comes to a halt
  - an extension of the *new* default theory is selected
  - and the reasoner resumes

How Did Reiter Define Extensions?

- Consider $W = \{a\}$ and D that consists of these two defaults

$$\frac{a : M\neg b}{c} \qquad \frac{a : M\neg c}{b}$$

- The only formulas that we stand any chance of deriving are

$$a,\ b,\ \text{and}\ c$$

- Since a is given, possible theories justifiable from (D, W) are

$$Cn(a),\ Cn(a, b),\ Cn(a, c),\ \text{and}\ Cn(a, b, c)$$

Extensions, Cont'd

- *Commit* to a candidate, say E
- This commitment allows us to interpret

  "is consistent to assume"
- G is consistent with a theory E if $E \not\models \neg G$
- That eliminates some rules as not usable, and removes justifications from the remaining ones
- Thus, leaving us with W and "monotone" inference rules
- Is what we adopted sensible? Does it allow us to reconstruct precisely what we adopted?

Extensions, Cont'd

- *It can be checked!*
- Consider $Cn(a, b)$

$$\frac{a : M\neg b}{c} \qquad \frac{a : M\neg c}{b} \qquad \Rightarrow \qquad \frac{a}{b}$$

  and it checks out!
- And so does $Cn(a, c)$; the other two do not
- *Extensions* are those candidate theories that make sense in this way
- The reasoner picks one and runs with it

With This Semantics Default Logic ...

- Could capture key commonsense reasoning phenomena
- Gave hope of conciseness of representation and of computational advantages
- Became a primary example of a *nonmonotonic logic*
- And a close relative to several other nonmonotonic logics
  - modal nonmonotonic logics of McDermott and Doyle, 1980, and McDermott 1982
  - autoepistemic logic modal logic of Moore, 1984 and 1985

Briefly About Nonmonotonicity in DL

- Localized to the stage when a reasoner is forced to commit to an extension
  - how to do it is another story
  - something as simple as gaining new information might require major changes in the extension
- Commitment to an extension a key feature
  - separates the default logic of Reiter from circumscription and *skeptical* versions of default logic and modal NMLs
- Other than that the process of reasoning *with* a default theory is classical

Where Did It Lead?

- The most important paper to shape my (and Victor's) views on nonmonotonic reasoning
- It was all there

Mostly in Parallel ...

- Another crucial line of research has been developing since late 1950s
- Could logic be used as a programming language?
  - Logic Theory Machine (Newell, Simon, Shaw, 1956) — a deduction system in propositional logic
  - Davis-Putnam-Logemann-Loveland *satisfiability* checking (1962)
  - Resolution method (Robinson 1965) where *unification* used to improve search

Indeed, Many Researchers Thought So

- PLANNER (Hewitt 1969) and its descendants
  - important because of THNOT
- Prolog (Colmerauer 1973, Roussel 1975)
  - an implementation of resolution for solving problems encoded as *queries* to a theory (logic program)
  - negation in the bodies of rules
- Logic Programming (Kowalski 1974)
  - Predicate logic [] a language for man-machine communication
  - The only [programming] language which is entirely user-oriented
- Predicative programming (Bibel 1975)
  - predicative program — definition part [given] in a language close to a natural one, and a control part generated automatically by machine and for machine

Quite a Different Focus

- Commonsense reasoning of no importance as a motivation
  - negation not a true negation
  - but that reflected computational considerations
- However, clearly useful for KR
  - declarative representations of objects and relations between them
  - modeled knowledge encapsulated in *(inductive) definitions*
- Inductive definitions in default logic? In other nonmonotonic logics?
  - not mentioned
- KR without definitions?
  - perhaps not viewed as an aspect of commonsense reasoning
  - but definitions and commonsense reasoning are closely related

About Logic Programs Then

- Logic program — a collection of logic program *rules*

$$a \leftarrow b_1, \ldots, b_m, \mathit{not}\ c_1, \ldots, \mathit{not}\ c_n$$

- When no negation — the meaning is clear
  - given by the least Herbrand model of the program
  - ground atoms in the model are true (they have a proof)
  - all others are false (by CWA)
- But negation is a “must have”
- Procedural implementations of the negation
  - first in PLANNER, then in Prolog
  - based on the idea of finite failure
- A declarative account of the negation became a key challenge

Negation in Logic Programming

- Clark's completion (Clark 1978)
- Stratification (Chandra and Harel, 1985; Apt, Blair Walker, 1986; Przymusinski, 1986)
- Default extension semantics for programs (Bidoit and Froidevaux, 1987)
- Autoepistemic expansion semantics for programs (Gelfond, 1987)
- Stable model semantics (Gelfond and Lifschitz, 1988)
- "Felicitous" model semantics (Fine, 1989)
- The well-founded semantics (Van Gelder, Ross and Schlipf, 1988)

Negation as Failure, Clark 1978

- The data comprises positive facts (truths) only
- A relation instance is false if we fail to prove that it is true
- Calls this negation *negation as failure*
  - likely Clark used the term first

Programs as Definitions, Clark 1978

- The assumption of *completeness of information*
  - an instance of a relation is true *only if* it is given explicitly or else is implied by one of the general rules for the relation
- The clauses that appear in a logic data base comprise *just the if-halves* of a set of if-and-only-if definitions, the only-if half of each definition being a *completion law* for the relation
- Problem with programs such as

$$p \leftarrow q$$
$$q \leftarrow p$$

- Hierarchical constraint

Definitions

- A well-understood linguistic construct used to specify a certain type of knowledge
- Often have two parts:
  - a list of objects with a property (that we are defining)
  - a set of rules that describing conditions under which additional objects have the property
  - a statement (often implicit) that unless an object can be shown to have the property using the rules above, it does not have this property
- Often inductive
- Often used in formal disciplines (mathematics)
  - typically in natural language, nevertheless unambiguous and precise
- But also in informal communication

Definitions

- 0 is an even integer
- If n is an even integer, so is $n + 2$
- (and no other integer is even)

Definitions

- No problem applying a definition if no negation in the body
- But what if some rules contain it?
- Applying such a rule may be problematic
  are we sure we can safely do so?

- We have a sense that only those definitions are correct that do not leave room for ambiguity as for when a rule can be used
- Inductive definitions that we all know in mathematics are all crystal clear in this respect

Stratification

- Partitions a program into layers (strata)
- Each layer is informed by lower layers; *never conversely*
- In particular, the negative information from lower layers “reduces” the layer to a Horn program
- The layer is then evaluated (with CWA used as in the case of Horn programs)
- Yields a *unique intended model* that defines the meaning of the program
- A minimal but not necessarily a least Herbrand model

- In the definition perspective: infer negative atoms when it is *safe*

Bidoit and Froidevaux, 1987

- Reiter's default logic is a well-suited formalism for defining the meaning of [] logic programs in a declarative fashion
- It provides a natural interpretation of negation, that we call *negation by default*
- Negation by default can be viewed as the declarative analog of negation by failure (Clark, 1978) which is essentially a computational notion

Bidoit and Froidevaux, 1987

- Program rules are defaults

$$a \leftarrow b_1, \ldots, b_k, not\ c_1, \ldots, not\ c_m$$

$$\frac{b_1 \wedge \ldots \wedge b_k : M\neg c_1, \ldots, M\neg c_m}{a}$$

- And so, logic programs are default theories
- Extensions of the default theory obtained from a program yield a class of models of the program
  - atoms in the extension are true in the model; others are false
- One of the two *first* definitions of stable models

Gelfond, 1987

- Program rules as modal formulas

$$a \leftarrow b_1, \ldots, b_k, not\ c_1, \ldots, not\ c_m$$

$$b_1 \wedge \ldots \wedge b_k \wedge \neg Lc_1 \wedge \ldots \wedge \neg Lc_m \supset a$$

- Programs are autoepistemic theories with the semantics of expansions (Moore 1984)
- The other of the two *first* definitions of stable models

Gelfond and Lifschitz, 1988

- An interpretation M is a stable model of a logic program P if it is the least model of its *reduct* with respect to M
  - *Gelfond-Lifschitz reduct* commonly written as P^M
- Remove each rule

$$a \leftarrow b_1, \ldots, b_k, \mathit{not}\ c_1, \ldots, \mathit{not}\ c_m$$

  with some c_i in M
- Remove all expressions *not c* from the bodies of all remaining rules
- What remain is the Gelfond-Lifschitz reduct

Gelfond and Lifschitz — Informal Reading

- Autoepistemic interpretation inserts the "belief" operator L after each negation

$$b_1 \wedge \ldots \wedge b_k \wedge \neg Lc_1 \wedge \ldots \wedge \neg Lc_m \supset a$$

- This mapping allows us to think about "negation as failure" in AEL
- *not B* expresses that the program gives no grounds for believing B
- Programs are interpreted *relative* to an agent
- They are about this agent's beliefs (knowledge)
- And so, we arrive at an *epistemic* view on the meaning of programs

Gelfond and Lifschitz — Informal Reading

- If M is the set of atoms that I consider true, then any rule that has a subgoal *not* B with $B \in M$ is, from my point of view, useless; furthermore, any subgoal *not* B with $B \notin M$ is, from my point of view trivial
- Then I can simplify the program and replace it by its reduct
- If M happens to be precisely the set of atoms that logically follow from the reduct, then I am *rational*
- But no reference to belief here
- They could have simply said "then M is a "good" model of the program
- Care needed when talking about the *informal semantics*

That's Where We Where in Early 1990s

- A handle on commonsense reasoning phenomena via nonmonotonic logics
- Solid understanding of programs with negation
- And of the interrelations between different nonmon formalisms
  - DL of vs AEL (Konolige 1987, M&T 1989, KR 1)
  - DL vs modal non-monotonic logics of McDermott and Doyle (Marek, Schwarz, Lifschitz, T, Lin and Shoham)
  - Nonmonotonninc logics and argumentation (Bondarenko, Kowalski and Toni, 1993)
  - Abductive logic programming (Kakas, Kowalski, and Toni, 1992, Denecker and De Schreye, 1992)
- But also, a sense of urgency around computational issues
  - promised (expected) computational efficiency
  - could it be verified and how?

Where Did It Lead?

- Existence of expansions in AEL (also algorithms for computing expansions (Ilkka Niemelä, 1988, 1992)
- Existence of extensions for literal-based normal and semi-normal default theories (Kautz and Selman, 1989, KR 1)
- Existence of expansions for the LP fragment of AEL (M&T 1989, 1991)
- Existence of extensions for general default theories (Stillman, 1992; Gottlob 1992)
- All eclipsed by results by Gottlob and Eiter on the complexity of abductive reasoning, knowledge base revision, disjunctive logic programming and more

Where Did It Lead?

- The Helsinki Logic Machine (around 1986-87)
  - the first research project that Ilkka Niemelä was involved in
  - included an implementation of some nonmonotonic reasoning tasks
- Default Reasoning System and TheoryBase — two projects at the University of Kentucky (around 1991-1992)

DeReS

- First implementation of (propositional) default logic (workshop in Shakertown, KY, in 1994, KR paper in 1996)
- Search for extensions via search for a set of *generating defaults*
- Build around the notion of splitting default theories
  - motivated by stratification of logic programs
  - developed by Pawel Cholewinski
  - closely related to splitting of logic programs by Lifschitz and Turner
- Slow because of
  - the large size of the search space
  - the cost of propagation during search
  - the use of tools for general propositional provability
- But it was a working proof of concept

TheoryBase

- To experiment with DeReS we needed default theories going beyond flying birds, employed adults, etc.
- Our proposal

  encode hard combinatorial problems as default theories with extensions as problem solutions

- Such problems abound and are of practical interest
- Could lead to practical applications of default logic

TheoryBase

- TheoryBase was a system to generate such theories
- Used Stanford GraphBase (Knuth) as the engine for generating underlying combinatorial structures
- Combined them with default encodings of problems to produce *propositional* default theories
- The basic methodology: *generate* a space of candidates; use *killing defaults* as constraints to eliminate unwanted ones
- The problems supported included:
  - maximal independent sets, maximal matchings, graph colorings, hamiltonian cycle, kernels
- We missed the idea of grounding

Lparse/Smodels

- Niemelä and Simons did it right!
- Focused on a fragment of DL - the *logic programming fragment*
- Focused on programs with variables
- Implemented grounding as a tool to produce propositional theories
- Implemented a DPLL-style solver
- Had first implementations around 1996
- Gave an impressive demo at LPNMR 1997

- A direct ancestor of all successful ASP systems

DIv

- Developed in Vienna and Calabria (Gottlob, Eiter, Leone)
- Soon after Lparse/smodels
- First announced at LPNMR 1997
- Focus on disjunctive logic programs
- Focus on grounding
- Connections with database query answering

With All That Going On

- It was time to step back and reflect
- That led to the two papers that verbalized the ASP pardigm
- The name came a year later, proposed by Vladimir Lifschitz

Inductive Definitions

- Hanging around but not given a central place
- But then, things changed
- Fine, 1989
- Denecker, 1998 and on

Fine's Insights

- Any program can be viewed as a *mechanism for generating truths:*
- To make rules with negation usable, in advance of the generation procedure, [we] *make a hypothesis* as to which statements are false
- Then this hypothesis can be used as part of the procedure to *detach negative statements* from the bodies
- The hypothesis is *happy* if it leads neither to *gaps* not *gluts*
- A model is *felicitous* if it embodies a happy hypothesis
- Explains negation as failure in terms of the *felicity* of models

Fine's Extended Inductive Definitions

- Fine recognizes that positive programs capture monotone inductive definitions
- But then asks: what if some clauses contain occurrences of negation within the body?
- Calls programs *extended inductive definitions*
- Defines a model to be *inductively defined* by a program when it results from the generation process

- Thus, a model is *inductively defined* if and only if it is felicitous (stable)

Inductive Definitions — Denecker

- The view that programs are inductive definitions was also made by Denecker in 1998
- Denecker (and later joined by Ternovska, Vennekens and Bogarts) studied the *process* of an inductive definition
- They argued that our understanding of definitions is captured by the well-founded semantics
- Embodies the principle

  establish negated literals if and when it is safe to do so

The Well-founded Semantics

- Proposed by Van Gelder, Ross and Schlipf in 1988
- An attempt to preserve the view of a single intended model of a program
- The idea: derive as much possible from the program and the CWA without introducing inconsistency
- *Unfounded* sets
- Required considering three-valued models
- Proved sound and became highly successful
- Correctly handled stratified programs

Denecker's proposal

- Definitions are essential for KR
- Definitions can be expressed as programs (possibly with negation)
- A definition is well-formed if its well-founded model is *total*

- Definitions may include open predicates
  - edge predicate in the definitions of the transitive closure
- A parameterized version of the wfs is needed

- If wfs is total then it is stable
- On correct definitions, the two semantics coincide

The Logic FO(ID) — Denecker

- The language of FO extended by *definitions*
- A formalism for knowledge representation
  - definitions can capture commonsense phenomena
  - e.g, reasoning about action in FO(ID)
- Modularity — theories can be analyzed one formula or definition at a time
  - formulas and definitions conjoined by the standard conjunction
- Intuitive, compelling informal semantics based on modularity and our common understanding of definitions

And finally — Informal Semantics

- Whatever else a formalism may be, at least some of its expressions must have referential semantics if the formalism is really to be a representation of knowledge
- That is, there must be some sort of correspondence between an expression and the world, such that it makes sense to ask whether the world is the way the expression claims it to be *(Moore, 1982)*

Informal Semantics in KR and LP

- Was often referred to

  a rule (default, etc) is informally read as ...

- But rarely taken seriously
- However, it should be taken seriously and carefully studied

Informal Semantics Formally

- In KR, a human expert models *informal propositions* about the domain of discourse by formal expressions in some logic $\mathcal{L}$
- The formal expressions are over a vocabulary for which the expert has an intended interpretation specifying the meaning of the vocabulary symbols in the domain
- The expert must understand which informal propositions about the domain are expressed by formal expressions of $\mathcal{L}$
- The informal semantics of $\mathcal{L}$ provides this understanding by explaining formal expressions of $\mathcal{L}$ as *precise* informal propositions about the domain of discourse
- Essential for modeling knowledge and declarative programming

Denecker, M., Lierler, Y., Truszczynski, M., and Vennekens, J.
The Informal Semantics of Answer Set Programming: A Tarskian perspective.
ICLP Technical Communications (2012)

Informal Semantics in Natural Language?

- To most logicians [] trained in model-theoretic semantics, natural language was an anathema, impossibly vague and incoherent *(Barwise and Cooper, 1981)*
- "Pretend-it's-English" is perilously vague; hard to judge whether two English sentences have the same meaning *(Hayes, 1977)*

Informal Semantics in Natural Language?

- I cannot put a thought in the hands of my readers with the request that they should examine it from all sides. Something in itself not perceptible by sense, the thought is presented to the reader — *and I must be content with that* — wrapped up in a perceptible linguistic form *(Frege, Der Gedanke, 1918)*
- To us, the revolutionary idea [] is the claim that *natural language is not impossibly incoherent* [] but that large portions of its semantics can be treated by combining known tools from logic *(Barwise and Cooper 1981)*

Objective or Epistemic

- The view of the world modeled as an FO interpretation (classical or Tarskian view)
- Reasoning about what holds and what does not
- Formulas explained by NL statements about objects in the application domain and relations among them

- The view of the world modeled as a collection of possible worlds (an S5-model)
- Reasoning about what an agent believes or knows
- Formulas explained by NL statements about what an agent believes or knows

Epistemic Informal Semantics

- $a \leftarrow b, \mathit{not}\ c$: *a holds if b holds and the agent does not know c*
- Inherited from DL and AEL because of how stable-model semantics came about
- Involves the concept of an agent
- Refers to what that agent knows about her beliefs
- Gives an intuitive, simple meaning to its rules
- But the composition operator (what makes a program out of rules) is non-standard and involves the notion of "all that agent knows"
  (Developed in AEL by Moore, Lakemeyer, Levesque)
- Applied to the *entire* the program

Objective Informal Semantics

- Programs have structure
- Consist of *groups* of rules playing different roles
  - TheoryBase
  - Stratification and splitting

Generate-Define-Test (Lifschitz 2002)

- Generate: rules that *generate* the space of candidate models
- Define: rules that define auxiliary notions
- Test (filter out): rules that eliminate unwanted models (constraints)

generate	$\{In(x,y)\} \leftarrow Edge(x,y).$
define	$T(x,y) \leftarrow In(x,y).$ $T(x,y) \leftarrow T(x,z), T(z,y).$
test	$\leftarrow In(x,y), In(x,z), y \neq z.$
	$\leftarrow In(x,z), In(y,z), x \neq y.$
	$\leftarrow Node(x), Node(y), not\ T(x,y).$

- Dominant programming methodology
- Modular structure, modules conected by *and*

Objective Informal Semantics of GDT Programs

- Builds on the standard informal semantics of FO logic
- Constraints interpreted as standard FO formulas
  - and so, have a standard reading we apply to FO
- Choice rules interpreted as standard FO formulas
  - and so, can be given a standard reading imported from FO
- *Define* modules explained as *definitions*
  - "all the agent knowns" vs "and nothing else is"
  - negation in definitions interpreted classically
- Close connection between the GDT fragment of ASP and FO(ID)
  - the wfs and the stable semantics coincide on correct definitions

(Denecker, Lierler, T, and Vennekens)

Objective or Epistemic

- Either provides a sound view of the program
- The choice depends on the application
- Epistemic view aligned with applications involving incomplete knowledge, commonsense reasoning, agents and their beliefs
- Objective view aligned with constraint solving applications of ASP

And That's the End

- What happened since then?
- And where are we going?

After 2000

- Aggregates (Niemelä, Simons and Soininen, 2002)
- Strong and uniform equivalence (Lifschitz, Pearce, Valverde; Eiter and Fink)
- Loop formulas (Lin and Zhao)

- ASP Programming Contests
- ASP 2.0 language standard and a new generation of tools
  - Gringo/clasp (Potsdam)
  - FOID (Leuven)
  - Asptools (Aalto)
  - Dlv and Wasp (Calabria)

- But also
  - C. Baral, Knowledge representation, reasoning and declarative problem solving, 2003
  - M. Gelfond and Y. Kahl, Knowledge Representation and the Design of Intelligent Agents; the ASP Approach, 2014

The Future

- Further improvement of modeling and solving tools
  - automated program optimization

- Integrating ASP with other AI methodologies to tackle complex applications requiring knowledge representation and reasoning, sensing and action, interaction between entities (agents), and learning

Wrapping Up

- A surprising confluence of ideas
- True to its roots in logic programming, ASP is now primarily a formalism for solving search and optimization problems
- True to its roots in commonsense reasoning and nonmonotonic logic, also a knowledge representation language for applications where agents, beliefs, and defaults are involved
- Along the way, the meaning of the default negation got clarified
- Inductive definitions were explained and gained a central role
- Informal semantics was untangled into two clear lines

Default negation, definitions, and informal semantics
They all came together!

Thank you!

References

Brewka G., Eiter T., and Truszczynski M. *Answer Set Programming at a Glance. Commun. ACM 54(12): 92-103 (2011)*

Marek V.W., Truszczynski M. *Stable Models and an Alternative Logic Programming Paradigm.* In: Apt K.R., Marek V.W., Truszczynski M., Warren D.S. (eds) The Logic Programming Paradigm. Springer (1990)

Niemelä, I. *Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm.* Ann. Math. Artif. Intell. 25(3-4): 241-273 (1999)

McCarthy, J. *Programs with Common Sense.* Symposium on Mechanization of Thought Processes. National Physical Laboratory, Teddington, England, 1958.

Newell, A. *The knowledge Level.* Presidential Address, AAAI 1980, AI Magazine 2(2) (1981).

Moore, R.C. *The Role of Logic in Knowledge Representation and Commonsense Reasoning.* AAAI: 428-433 (1982)

McCarthy, J. *Circumscription - A Form of Non-Monotonic Reasoning.* Artif. Intell. 13(1-2): 27-39 (1980)

Reiter, R., *A Logic for Default Reasoning.* Artif. Intell. 13(1-2): 81-132 (1980)

McDermott, D. and Doyle, J. *Non-Monotonic Logic I.* Artif. Intell. 13(1-2): 41-72 (1980)

References

McDermott, D.V. *Nonmonotonic Logic II: Nonmonotonic Modal Theories.* J. ACM 29(1): 33-57 (1982)

Newell, A. and Simon, H. *The logic theory machine–A complex information processing system.* IRE Transactions on information theory, 1956

Davis, M., Logemann, G., and Loveland, D. *A machine program for theorem-proving.* Communications of the ACM, 1962

Robinson, J.A. *A Machine-Oriented Logic Based on the Resolution Principle.* J. ACM 12(1): 23-41 (1965)

Hewitt, C. *PLANNER: A Language for Proving Theorems in Robots.* IJCAI, 295-302, 1969

Colmerauer, A. and Roussel, P., *The Birth of Prolog*, In: History of Programming languages—II, eds., Bergin,Jr. T.J. and Gibson,Jr. R.G., ACM, 1996

Kowalski, R.A. *Predicate Logic as Programming Language.* IFIP Congress 1974: 569-574

Bibel, W. *Prädikatives Programmieren.* Automata Theory and Formal Languages, 274-283, 1975

Clark, K.L. *Negation as Failure.* Logic and Data Bases, 293-322, 1978

Apt, K., Blair, H.A., and Walker, A. *Towards a theory of declarative knowledge.* In: Foundations of deductive databases and logic programming. Ed., Minker, J., Morgan-Kaufman, 1988.

References

Bidoit, N., and Froidevaux, C. *Minimalism subsumes default logic and circumscription.* LICS, 89-97, 1987

Gelfond, M. *On Stratified Autoepistemic Theories.* AAAI, 207-211, 1987

Gelfond, M., and Lifschitz, V. *The Stable Model Semantics for Logic Programming.* ICLP/SLP, 1070-1080, 1988

Fine, K. *The justification of negation as failure.* Proceedings of the 8th International Congress of Logic, Methodology and Philosophy of Science, North Holland, 1989

Van Gelder, A., Ross, K.A., and Schlipf, J.S. *Unfounded Sets and Well-Founded Semantics for General Logic Programs.* PODS, 221-230, 1988

Niemelä, I., and Simons, P. *Efficient Implementation of the Well-founded and Stable Model Semantics.* JICSLP, 289-303, 1996

Niemelä, I., and Simons, P. *Smodels - An Implementation of the Stable Model and Well-Founded Semantics for Normal LP* LPNMR, 421-430, 1997

Eiter, T., Leone, N., Mateis, C., Pfeifer, G., Scarcello, F. *A Deductive System for Non-Monotonic Reasoning.* LPNMR, 364-375, 1997

Aczel, P. *An introduction to inductive definitions.* In: Studies in Logic and the Foundations of Mathematics, Elsevier, 1977

Denecker, M. *The Well-Founded Semantics Is the Principle of Inductive Definition.* JELIA, 1-16, 1998

References

Denecker, M. *Extending Classical Logic with Inductive Definitions.* Computational Logic, 703-717, 2000.

Denecker, M., and Vennekens, J. *The Well-Founded Semantics Is the Principle of Inductive Definition, Revisited.* KR, 2014

- Barwise, J., and Cooper, R. *Generalized quantifiers and natural language.* Linguistics and Philosophy 4, 159-219, 1981
- Denecker, M., Lierler, Y., Truszczynski, M., and Vennekens, J. *The Informal Semantics of Answer Set Programming: A Tarskian Perspective.* ICLP Technical Communications, 2012