Programming Paradigms: Logic Programming

Motivation and Introduction

Víctor Gutiérrez Basulto

Based on slides available at `https://potassco.org/teaching/` (CC-BY) and partially on slides by Daria Stepanova, Max-Planck Institut für Informatik

30 • 09 • 2019

Programming Paradigms: Why to bother?

- Tools (programming languages) may not suffice to deal with *all* real life problems
- Learning programming languages becomes easier
- Collaborate in the development of programming technologies

One step back: Programming Languages

- A programming language is

One step back: Programming Languages

- A programming language is an artificial language designed to communicate instructions to a machine e.g./ computer
- Programming languages provide an abstraction from a computer's instruction set architecture
- Low-level programming languages provide little or no abstraction, machine code and assembly language
- High-level programming languages isolate the execution semantics of a computer architecture from the specification of the program
  - Simplifies program development

Programming Paradigms

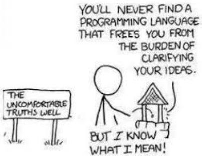
- Programming languages can be categorized into programming paradigms

Programming Paradigms

- Programming languages can be categorized into programming paradigms
- Meaning of the word ‘paradigm’
  - “An example that serves as pattern or model” - *The American Heritage Dictionary of the English Language*
  - “Paradigms emerge as the result of social processes in which people develop ideas and create principles and practices that embody those ideas” - *Thomas Kuhn, “The Structure of Scientific Revolutions”*
- Programming paradigms are the result of people’s ideas about how computer programs should be constructed
  - Patterns that serves as a “school of thoughts” for programming of computers

Programming Paradigms/2

- AGAIN: Once you have understood the general concepts of programming paradigms, it becomes easier to learn new programming languages
- However, this does not mean that by just picking the right paradigm all problems vanish into thin air


YOU'LL NEVER FIND A PROGRAMMING LANGUAGE THAT FREES YOU FROM THE BURDEN OF CLARIFYING YOUR IDEAS.
THE UNCOMFORTABLE TRUTHS WELL
BUT I KNOW WHAT I MEAN!

- Or put more elegantly:

> “There does not now, nor will there ever exist, a programming language in which it is the least bit hard to write bad programs.” - L. Flon

Principal Programming Paradigms (One possible classification)

- Imperative
  - Procedural
  - Object Oriented
- Declarative
  - Functional Programming
  - Logic Programming
- Scripting
- Concurrent
- Assambler

Principal Programming Paradigms (One possible classification)

- Imperative
    - Procedural
    - Object Oriented
- Declarative
    - Functional Programming
    - Logic Programming
- Scripting
- Concurrent
- Assambler

Principal Programming Languages: Two Key Aspects

- Syntax of Language:

Principal Programming Languages: Two Key Aspects

- Syntax of Language: describes how well formed expression should look like
- For example, the following (English) sentence is not correct:

  “Furiously slqxp ideas grn colorless”
- In contrast, the sentence

  “Colorless green ideas sleep furiously”

  is syntactically correct (but it does not make any sense).

Principal Programming Languages: Two Key Aspects/2

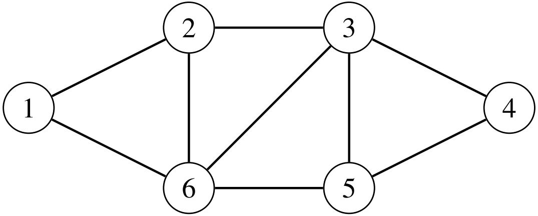
- Semantics:

Principal Programming Languages: Two Key Aspects/2

- Semantics: is concerned with the meaning of (programming) languages
  - normally much more difficult than the syntax
- A programmer should "*anticipate*" what will happen before actually running the programm
- An accurate description of the meaning of language constructs is needed.

Logic Programming: Answer Set Programming

3 Graph-Coloring



- TASK
  - Color the nodes of the graph in three colors such that no two adjacent nodes share the same color.

Sudoku

	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
		6				3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

- TASK
  - Fill in the grid so that every row, every column, and every 3x3 box contains the digits 1 through 9.

Declarative Language

- We explicitly represent knowledge to describe what the problem is.
- Example: Declarative description of a number of being prime

Declarative Language

- We explicitly represent knowledge to describe what the problem is.
- Example: Declarative description of a number of being prime

prime(n): ($n > 1$) and for all m divide(m,n) implies $m = 1$ or $m = n$

Central Ideas

- Goal:
  - We aim for a computational approach for declarative problem solving.
- Declarative:
  - There is a separation between the representation of knowledge and the processing of knowledge

Answer Set Programming

- Answer Set Programming (ASP) is a recent problem solving approach, based on declarative programming.
- The term was coined by Michael Gelfond and Vladimir Lifschitz [1999,2002]
- It has roots in knowledge representation, logic programming, and nonmonotonic reasoning.
- At an abstract level, ASP relates to SAT solving and constraint satisfaction problems (CSPs).

- ASP is an approach to declarative problem solving, combining
  - a rich yet simple modeling language
  - with high-performance solving capacities

  tailored to Knowledge Representation and Reasoning
- ASP has its roots in
  - deductive databases
  - logic programming
  - logic-based knowledge representation
  - constraint solving (in particular, SATisfiability testing)
- ASP allows for solving all search problems in NP (and NP^{NP}) in a uniform way

ASP Systems

ASP has gained importance in several areas of computer science

- High Expressivity
- Several Solvers
  - DLV (TU Wien, University of Calabria)
  - clasp (University of Potsdam)
  - Smodels, GnT (Aalto University)
  - ASSAT (Hong Kong University of Science and Technology)

    .
    .
    .

⇒ DLV and clasp most efficient at the present

ASP Systems

ASP has gained importance in several areas of computer science

- High Expressivity
- Several Solvers
  - DLV (TU Wien, University of Calabria)
  - clasp (University of Potsdam)
  - Smodels, GnT (Aalto University)
  - ASSAT (Hong Kong University of Science and Technology)

    .
    .
    .

⇒ DLV and clasp most efficient at the present

Central Ideas (cont'd)

ASP is a declarative method to represent and solve problems based on the following methodology

- Problems are represented in terms of (ASP are)logic programs with a semantics such that it holds:
  - Solutions of a given problem are determined by models (answer sets or stable models) of the logic program.
  - Answer sets are a "selection" of all classical models.
- Fundamental characteristics:
  - models, not proofs, represent solutions; requires techniques to compute models (rather than techniques to compute proofs)

ASP Applications

- Combinatorial search problems (some with substantial amount of data), like

ASP Applications

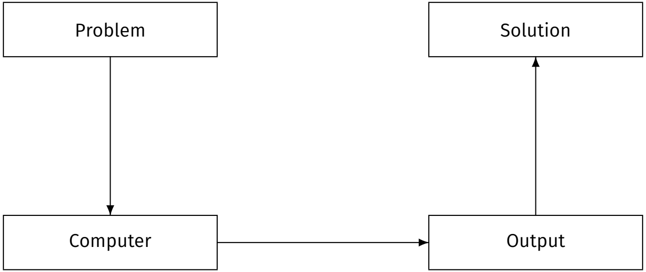
- Combinatorial search problems (some with substantial amount of data), like
  - Automated planning
  - Constraint Satisfaction,
  - Information integration
  - Diagnosis, Repair
  - Music composition
  - Product configuration
  - Robotics
  - System design
  - Systems biology
  - and many many more

See AI Magazine article on ASP [Erdem et al., 2016] for overview

ASP applications

- **USA-Advisor** [Nogueira et al., 2001]
  - decision support system to control the Space Shuttle during flight
  - issue: problems with the oxygen transport (pipes and valves)
  - failure scenario: also multiple system failures occur
- **Biological Network Repair** [Kaminski et al., 2013]
  - model nodes (substances, etc) in a large scale biological influence graph, with roles (e.g. inhibitor, activator)
  - repair inconsistencies (modify roles, add links between nodes, etc)
- **Anton** [Boenn et al., 2011] http://www.cs.bath.ac.uk/ mjb/anton/
  - automatic system for the composition of renaissance-style music.
  - musical knowledge ? 500 ASP rules (melody, harmony, rhythm) ?? can generate musical pieces, check pieces for violations.

ASP Approach in a Nutshell



Problem

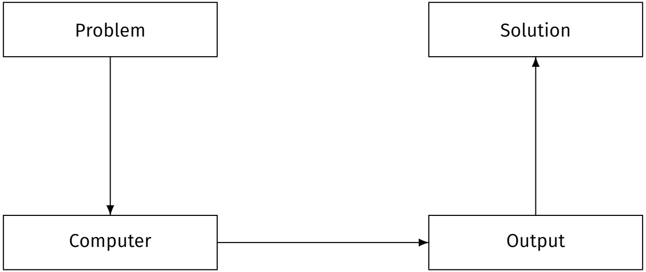
Computer

Output

Solution

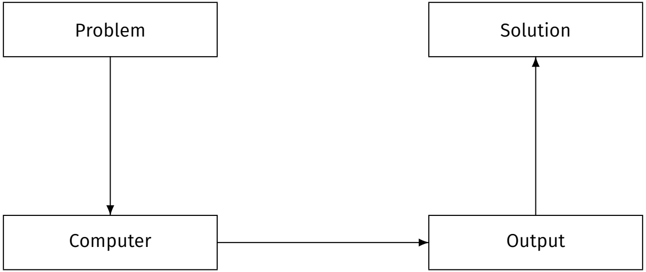
ASP Approach in a Nutshell

“What is the problem?” versus “How to solve the problem?”



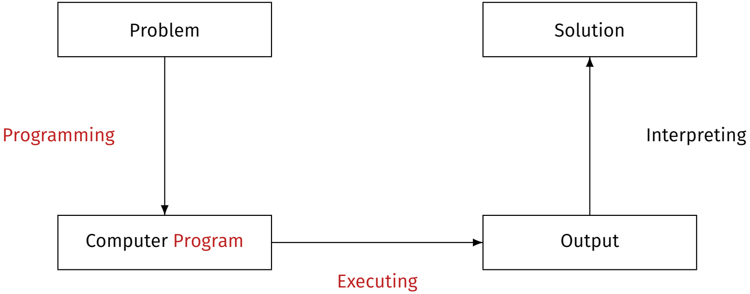
ASP Approach in a Nutshell

“What is the problem?” versus “How to solve the problem?”



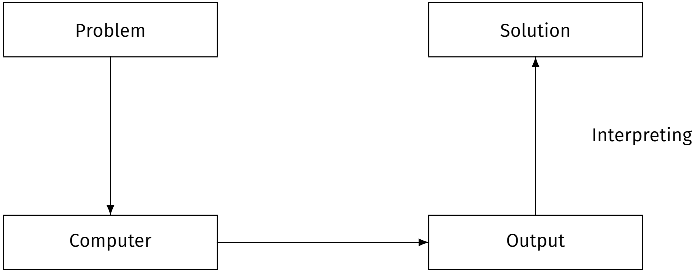
ASP Approach in a Nutshell

“What is the problem?” versus “How to solve the problem?”



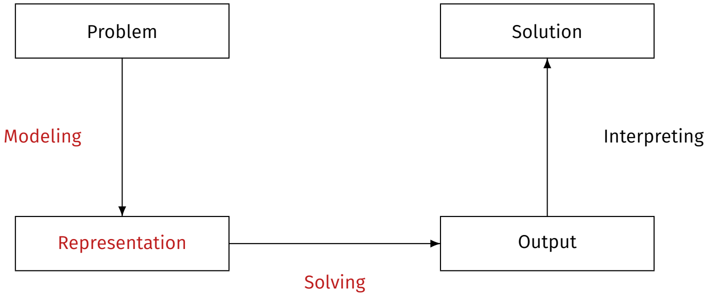
ASP Approach in a Nutshell

“What is the problem?” versus “How to solve the problem?”

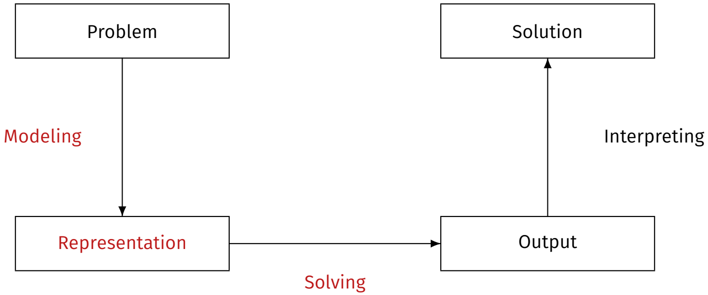


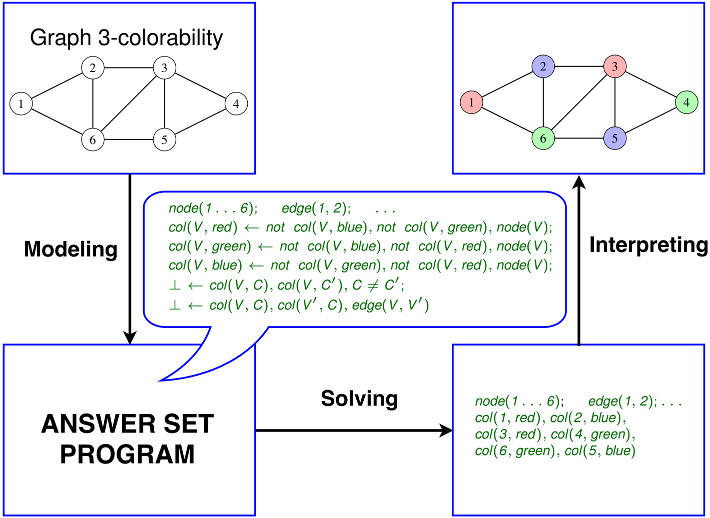
ASP Approach in a Nutshell

“What is the problem?” versus “How to solve the problem?”



ASP Approach in a Nutshell





Graph 3-colorability

Modeling

$node(1 \ldots 6); \quad edge(1,2); \quad \ldots$

$col(V, red) \leftarrow not\ \ col(V, blue), not\ \ col(V, green), node(V);$

$col(V, green) \leftarrow not\ \ col(V, blue), not\ \ col(V, red), node(V);$

$col(V, blue) \leftarrow not\ \ col(V, green), not\ \ col(V, red), node(V);$

$\bot \leftarrow col(V, C), col(V, C'), C \neq C';$

$\bot \leftarrow col(V, C), col(V', C), edge(V, V')$

Interpreting

ANSWER SET PROGRAM

Solving

$node(1 \ldots 6); \quad edge(1,2); \ldots$

$col(1, red), col(2, blue),$

$col(3, red), col(4, green),$

$col(6, green), col(5, blue)$

Paradigm shift

Model Generation based approach (eg. SATisfiability testing)

1. Provide a representation of the problem
2. A solution is given by a model of the representation

➥ Answer Set Programming (ASP)

ASP-style playing with blocks

Logic program

```
on(a,b).
on(b,c).

above(X,Y) :- on(X,Y).
above(X,Y) :- on(X,Z), above(Z,Y).
```

ASP-style playing with blocks

Logic program

```
on(a,b).
on(b,c).

above(X,Y) :- on(X,Y).
above(X,Y) :- on(X,Z), above(Z,Y).
```

Stable Herbrand model

$$\{\ \mathtt{on}(\mathtt{a},\mathtt{b}),\ \mathtt{on}(\mathtt{b},\mathtt{c}),\ \mathtt{above}(\mathtt{b},\mathtt{c}),\ \mathtt{above}(\mathtt{a},\mathtt{b}),\ \mathtt{above}(\mathtt{a},\mathtt{c})\ \}$$

ASP-style playing with blocks

Logic program

```
on(a,b).
on(b,c).

above(X,Y) :- on(X,Y).
above(X,Y) :- on(X,Z), above(Z,Y).
```

Stable Herbrand model (and no others)

$$\{ \texttt{on}(a,b),\ \texttt{on}(b,c),\ \texttt{above}(b,c),\ \texttt{above}(a,b),\ \texttt{above}(a,c) \}$$

- A logic program P is a **set of rules** of the form

$$\underbrace{a}_{\text{head}} \leftarrow \underbrace{b_1, \ldots, b_m, \neg c_1, \ldots, \neg c_n}_{\text{body}}$$

  - a and all b_i, c_j are atoms (propositional variables)
  - $\leftarrow$, $,$, $\neg$ denote if, and, and negation
  - intuitive reading: head must be true if body holds

- A logic program P is a **set of rules** of the form

$$\underbrace{a}_{\text{head}} \leftarrow \underbrace{b_1, \ldots, b_m, \neg c_1, \ldots, \neg c_n}_{\text{body}}$$

  - a and all b_i, c_j are atoms (propositional variables)
  - $\leftarrow$, ,, $\neg$ denote if, and, and negation
  - intuitive reading: head must be true if body holds

- Semantics given by stable models, informally,
  models of P justifying each true atom by some rule in P

- ASP as High-level Language
  - Express problem instance(s) as sets of facts
  - Encode problem (class) as a set of rules
  - Read off solutions from stable models of facts and rules

- ASP as Low-level Language
  - Compile a problem into a logic program
  - Solve the original problem by solving its compilation

Mandatory Reading for next session: Chapter 1 and Section 2.1 of Answer Set Solving in Practice, by Gebser, Kaminski, Kaufmann, Schaub