

Map Coloring Problem with *Answer Set Programming – clingo* An Encoding

Claudio Cesar de Sá

Independent Researcher

Road map of this presentation:

1. About the ASP (clingo)
2. Requisites
3. The problem: map coloring
4. Discussion of this (NP-complete) problem
5. A modelling in ASP
6. A solution in clingo
7. Conclusions

Road map of this presentation:

1. About the ASP (clingo)
2. Requisites
3. The problem: map coloring
4. Discussion of this (NP-complete) problem
5. A modelling in ASP
6. A solution in clingo
7. Conclusions

Attention: some background in logic and declarative language is recommended!

Answer Set Programming

in a Nutshell (this page was borrowed from official repository):

Answer Set Programming

in a Nutshell (this page was borrowed from official repository):

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

Answer Set Programming

in a Nutshell (this page was borrowed from official repository):

- ▶ ASP is an approach to **declarative problem solving**, combining
 - ▶ a rich yet simple modeling language
 - ▶ with high-performance solving capacities
- ▶ ASP has its roots in
 - ▶ (deductive) databases
 - ▶ logic programming (with negation)
 - ▶ (logic-based) knowledge representation and (non-monotonic) reasoning
 - ▶ constraint solving (in particular, SATisfatibility testing)

Answer Set Programming

in a Nutshell (this page was borrowed from official repository):

- ▶ ASP is an approach to **declarative problem solving**, combining
 - ▶ a rich yet simple modeling language
 - ▶ with high-performance solving capacities
- ▶ ASP has its roots in
 - ▶ (deductive) databases
 - ▶ logic programming (with negation)
 - ▶ (logic-based) knowledge representation and (non-monotonic) reasoning
 - ▶ constraint solving (in particular, SATisfatibility testing)
- ▶ ASP allows for solving all search problems in NP (and NP^{NP}) in a uniform way

Answer Set Programming

in a Nutshell (this page was borrowed from official repository):

- ▶ ASP is an approach to **declarative problem solving**, combining
 - ▶ a rich yet simple modeling language
 - ▶ with high-performance solving capacities
- ▶ ASP has its roots in
 - ▶ (deductive) databases
 - ▶ logic programming (with negation)
 - ▶ (logic-based) knowledge representation and (non-monotonic) reasoning
 - ▶ constraint solving (in particular, SATisfatibility testing)
- ▶ ASP allows for solving all search problems in NP (and NP^{NP}) in a uniform way
- ▶ ASP fundamentals in: propositional, first-order, auto-epistemic, default. Many aspects in theory and programs.

Answer Set Programming

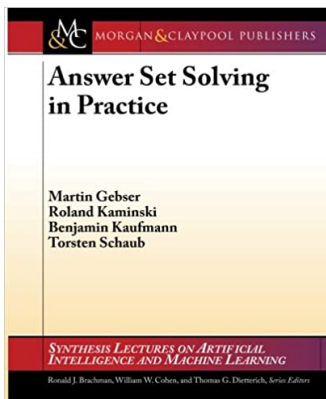
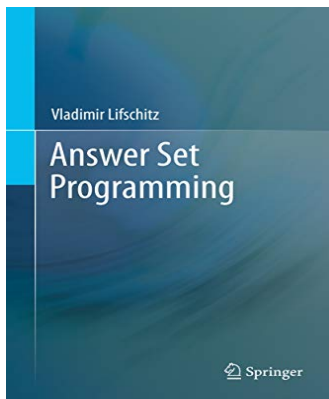
in a Nutshell (this page was borrowed from official repository):

- ▶ ASP is an approach to **declarative problem solving**, combining
 - ▶ a rich yet simple modeling language
 - ▶ with high-performance solving capacities
- ▶ ASP has its roots in
 - ▶ (deductive) databases
 - ▶ logic programming (with negation)
 - ▶ (logic-based) knowledge representation and (non-monotonic) reasoning
 - ▶ constraint solving (in particular, SATisfatibility testing)
- ▶ ASP allows for solving all search problems in NP (and NP^{NP}) in a uniform way
- ▶ ASP fundamentals in: propositional, first-order, auto-epistemic, default. Many aspects in theory and programs.
- ▶ ASP embraces many emerging application areas

Historic and references:

- ▶ This programming language has its origin at the University of Potsdam (Universität Potsdam) – 1999
- ▶ **Potassco**, the **P**otsdam **A**nswer **S**et **S**olving **C**ollection –
<https://potassco.org/>
- ▶ Official repository with a full-course:
<https://github.com/potassco-asp-course/>
- ▶ Support to start: an active forum and a course covered by videos in the Youtube
- ▶ This presentation and its code:
https://github.com/claudiosa/CCS/tree/master/asp_Answer_Set_Programming
- ▶ Books:

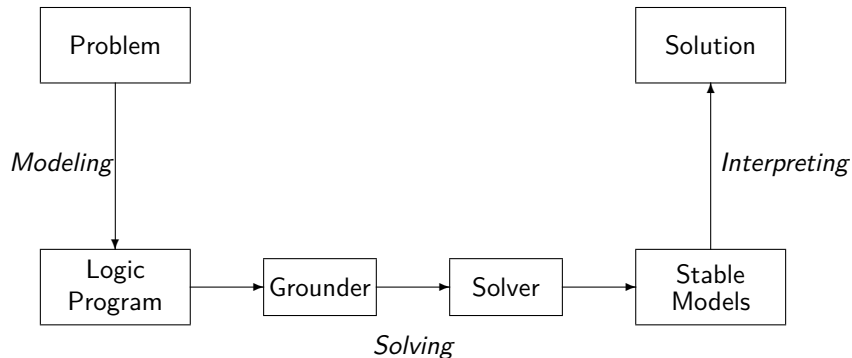
Some References:



Characteristics:

- ▶ The theoretical background has the roots from many logics: propositional, first-order, circumscription (world closed supposition), default and the negation as a fail.
- ▶ The concept of **stable model** to deduce one or more answers.
- ▶ The combinatorial problems are well modelled with ASP (*wear as glove*)
- ▶ The encoding declare **decisions** e **constraints**

Modelling, grounding and resolution:



Source: <https://github.com/potassco-asp-course/>

A case-study: Map Coloring Problem (a practical problem from graph coloring)

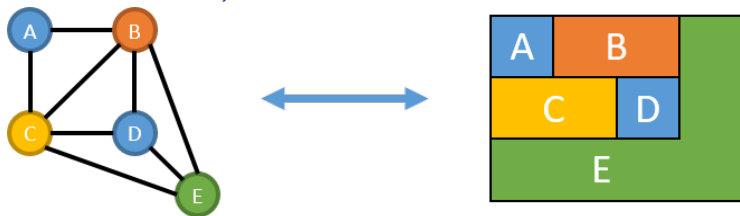


Figura: Let's consider a planar map for readability

- ▶ Input: a graph (map) G with n vertices (countries) and integer k (colors)
- ▶ Output: does G admit a proper vertex coloring with k colors?
- ▶ Complexity: NP-Complete
- ▶ Optimization: NP-Hard (lesser chromatic number – k)

Borrowed and adapted from:

<https://github.com/potassco-asp-course/modeling/>

- ▶ **Problem instance:** A map consisting of countries and neighborhood (frontiers)
 - ▶ facts formed by predicates `country/1` and `neighbour/2`
 - ▶ facts formed by predicate `color/1`
- ▶ **Problem class:** Assign each country one color such that no two countries connected or neighbour have the same color
- ▶ In other words,
 1. Each country has one color
 2. Two connected (neighbour) countries must not have the same color

Let's paint this map with k colors:



Figura: South America map, the author had the input data – 😊

Let's paint this map with k colors:



Figura: South America map, the author had the input data – 😊

Finally, let's try to find a minimal color number to paint this map, without any neighbour country have the same color!

Some comments:

- ▶ Theoretical details can be found in:
https://en.wikipedia.org/wiki/Graph_coloring
- ▶ The modelling was almost immediate with an old Prolog code from the author:
<https://github.com/claudiosa/CCS/.....>
- ▶ Many approaches for this problem can be taken: Simulated Annealing, Ant Colony, Depth-First Search, ..., meta-heuristics in general presents a good efficiency
- ▶ The full code discussed here is found in:
https://github.com/claudiosa/CCS/tree/master/asp_Answer_Set_Programming/map_coloring.lp
- ▶ We are commenting it in parts

Colors available (k) and countries (n):

```
%% Colors
color(red).
color(blue).
color(green).
color(yellow).
%% Countries
country(antilles).      country(argentina).
country(bolivia).       country(brazil).
country(columbia).      country(chile).
country(ecuador).       country(french_guiana).
country(guyana).         country(paraguay).
country(peru).           country(surinam).
country(uruguay).       country(venezuela).
```

Ground terms, exactly written like Prolog syntax.

The map, countries and their relations with neighbours:

```
neighbour(antilles,venezuela).    neighbour(argentina,bolivia).
neighbour(argentina,brazil).       neighbour(argentina,chile).
neighbour(argentina,paraguay).     neighbour(argentina,uruguay).
neighbour(bolivia,brazil).         neighbour(bolivia,chile).
neighbour(bolivia,paraguay).       neighbour(bolivia,peru).
neighbour(brazil,columbia).        neighbour(brazil,french_guiana)
neighbour(brazil,guyana).          neighbour(brazil,paraguay).
neighbour(brazil,peru).            neighbour(brazil,surinam).
neighbour(brazil,uruguay).         neighbour(brazil,venezuela).
neighbour(chile,peru).             neighbour(columbia,ecuador).
% To avoid duplication of the facts above
neighbour(X,Y) :- neighbour(Y,X).
% To obtain more symmetric results at the end
```

Another representation is possible, but until now, everything was reused – kept simple as possible!

Modelling the problem under its requisites:

```
%% Country X Colors - Assign any color for each country
1 { country_color(P, C) : color(C) } 1 :- country(P).

%% Brazil must be green
:- not country_color(brazil,green).
%% OR
%% country_color(brazil,green).

%% Finally: none adjacent countries receive at the same color
% C != C1 :- country_color(P, C), country_color(P1, C1),
           neighbour(P,P1).
%% OR -- by Susana - ASP Community
:- country_color(P, C), country_color(P1, C), neighbour(P,P1).
```

Basically, that's all!

Preparing for a optimization:

```
%% number of colors used
n_colors(N) :-    N = #count{C : country_color(P,C)}.

%% A minimization on this value
#minimize{ N : n_colors(N) }.

%% OUTPUT
#show country_color/2.
#show n_colors/1.
```

That's all!

An output:

```
clingo ../map_coloring.lp 0 --out-ifs='\n' --out-atomf=%s.
clingo version 5.3.0
Reading from ../map_coloring.lp
Solving...
Answer: 1
country_color(argentina,red).
country_color(columbia,red).
country_color(surinam,red).
country_color(guyana,blue).
country_color(paraguay,blue).
.....
country_color(french_guiana,yellow).
country_color(venezuela,yellow).
n_colors(4).
Optimization: 4
OPTIMUM FOUND
Models          : 1
  Optimum       : yes
Optimization    : 4
Calls           : 1
```

Conclusions:

- ▶ ASP is strongly declarative (roots from the logic to attack the problems representation)
- ▶ Generate and test methodology
- ▶ ASP's workflow, modeling, grounding, solving (and optimizing)
- ▶ `clingo` = `gringo`+`clasp` + ...
- ▶ Allows you to embed a Python coding in order to minimize the difficulties (☹) of input and output data
- ▶ An encoding in ASP is excellent exercise to keep your mind very active!
- ▶ Finally, a huge gratitude for the **potassco-users list**, always reactive for my silly doubts, where I had been learning much.

Contact and Comments (are must welcome 😊):

- ▶ <https://claudiocesars.wordpress.com/>
- ▶ This presentation and the code discussed:
https://github.com/claudiosa/CCS/tree/master/asp_Answer_Set_Programming
- ▶ There is a directory to Youtube!
- ▶ ✉: ccs1664@gmail.com
- ▶ This material has a partial support from WhatsTV Inc.
<https://en.whatstv.com.br/>, here our gratitude!
- ▶ *Thank you so much!*