Constraint Satisfaction Problems in Python

Michael Sioutis

Department of Informatics and Telecommunications
National and Kapodistrian University of Athens

July 18, 2011

Table of Contents

Definition

A Constraint Satisfaction Problem consists of:

- A Finite set of *variables*: $V_1, V_2, \ldots, V_n$
- A Nonempty *domain* of possible values for each variable: $D_{V_1}, D_{V_2}, \ldots, D_{V_n}$
- A Finite set of constraints: $C_1, C_2, \ldots, C_n$
  - Each constraint C_i limits the values that variables can take (e.g., $V_1 \neq V_2$)

Definition

A Constraint Satisfaction Problem consists of:

- A Finite set of *variables*: $V_1, V_2, \ldots, V_n$
- A Nonempty *domain* of possible values for each variable: $D_{V_1}, D_{V_2}, \ldots, D_{V_n}$
- A Finite set of constraints: $C_1, C_2, \ldots, C_n$
    - Each constraint C_i limits the values that variables can take (e.g., $V_1 \neq V_2$)

Definition

A Constraint Satisfaction Problem consists of:

- A Finite set of *variables*: $V_1, V_2, ..., V_n$
- A Nonempty *domain* of possible values for each variable: $D_{V_1}, D_{V_2}, ..., D_{V_n}$
- A Finite set of constraints: $C_1, C_2, ..., C_n$
  - Each constraint C_i limits the values that variables can take (e.g., $V_1 \neq V_2$)

Definition

- A *state* is defined as an assignment of values to some or all variables.
- A *consistent assignment* is an assignment that does not violate the constraints.
- A *complete assignment* is an assignment that includes all variables.
- A *problem solution* is a complete and consistent assignment.

Definition

- A *state* is defined as an assignment of values to some or all variables.
- A *consistent assignment* is an assignment that does not violate the constraints.
- A *complete assignment* is an assignment that includes all variables.
- A *problem solution* is a complete and consistent assignment.

Definition

- A *state* is defined as an assignment of values to some or all variables.
- A *consistent assignment* is an assignment that does not violate the constraints.
- A *complete assignment* is an assignment that includes all variables.
- A *problem solution* is a complete and consistent assignment.

Definition

- A *state* is defined as an assignment of values to some or all variables.
- A *consistent assignment* is an assignment that does not violate the constraints.
- A *complete assignment* is an assignment that includes all variables.
- A *problem solution* is a complete and consistent assignment.

Benefits

- Standard representation pattern
- Generic goal and successor functions
- Generic heuristics (no domain specific expertise)

Benefits

- Standard representation pattern
- Generic goal and successor functions
- Generic heuristics (no domain specific expertise)

Benefits

- Standard representation pattern
- Generic goal and successor functions
- Generic heuristics (no domain specific expertise)

Applications

- Scheduling the time of observations on the Hubble Space Telescope
- Airline schedules
- Map coloring
- Cryptography
- Scheduling your MS or PhD thesis exam ☺

Applications

- Scheduling the time of observations on the Hubble Space Telescope
- Airline schedules
- Map coloring
- Cryptography
- Scheduling your MS or PhD thesis exam ☺

Applications

- Scheduling the time of observations on the Hubble Space Telescope
- Airline schedules
- Map coloring
- Cryptography
- Scheduling your MS or PhD thesis exam ☺

Applications

- Scheduling the time of observations on the Hubble Space Telescope
- Airline schedules
- Map coloring
- Cryptography
- Scheduling your MS or PhD thesis exam ☺

Applications

- Scheduling the time of observations on the Hubble Space Telescope
- Airline schedules
- Map coloring
- Cryptography
- Scheduling your MS or PhD thesis exam ☺

Constraint Programming In Python Possible?

- Constraint satisfaction problems are mathematical problems defined as a set of objects whose state must satisfy a number of constraints or limitations.
- We only need to specify the problem, even better if we could do it in Python and make use of its powerful features...
- We can! With the *python-contraint*[1] module.

---

[1] http://labix.org/python-constraint

Constraint Programming In Python Possible?

- Constraint satisfaction problems are mathematical problems defined as a set of objects whose state must satisfy a number of constraints or limitations.
- We only need to specify the problem, even better if we could do it in Python and make use of its powerful features...
- We can! With the *python-contraint*[1] module.

---

[1] http://labix.org/python-constraint

Python-Constraint Library

- Python module *python-constraint* offers solvers for Constraint Satisfaction Problems over finite domains in simple and pure Python.
- Download and install *python-constraint* from here: http://labix.org/download/python-constraint/python-constraint-1.1.tar.bz2
- After you setup, you should be able to run the following command on a python shell:

```
from constraint import *
```

Python-Constraint Library

- Python module *python-constraint* offers solvers for Constraint Satisfaction Problems over finite domains in simple and pure Python.
- Download and install *python-constraint* from here: http://labix.org/download/python-constraint/python-constraint-1.1.tar.bz2
- After you setup, you should be able to run the following command on a python shell:

```
from constraint import *
```

Features Of Python-Constraint

- Solvers
    - Backtracking solver
    - Recursive backtracking solver
    - Minimum conflicts solver
- Predefined constraint types (e.g., AllDifferentConstraint, FunctionConstraint)

Features Of Python-Constraint

- Solvers
    - Backtracking solver
    - Recursive backtracking solver
    - Minimum conflicts solver
- Predefined constraint types (e.g., AllDifferentConstraint, FunctionConstraint)

Solving An Algebraic Relation

- Solve the $a + b = 5, a * b = 6$ algebraic relation.
-
```
from constraint import *
problem = Problem()
problem.addVariable('a', range(5))
problem.addVariable('b', range(5))
problem.addConstraint(lambda a, b: a + b == 5)
problem.addConstraint(lambda a, b: a * b == 6)
solutions = problem.getSolutions()
print solutions
```
-
```
[{'a': 3, 'b': 2}, {'a': 2, 'b': 3}]
```

Solving An Algebraic Relation

- Solve the $a + b = 5, a * b = 6$ algebraic relation.
-
  ```
  from constraint import *
  problem = Problem()
  problem.addVariable('a', range(5))
  problem.addVariable('b', range(5))
  problem.addConstraint(lambda a, b: a + b == 5)
  problem.addConstraint(lambda a, b: a * b == 6)
  solutions = problem.getSolutions()
  print solutions
  ```
-
  ```
  [{'a': 3, 'b': 2}, {'a': 2, 'b': 3}]
  ```

Solving An Algebraic Relation

- Solve the $a + b = 5, a * b = 6$ algebraic relation.
-
```
from constraint import *
problem = Problem()
problem.addVariable('a', range(5))
problem.addVariable('b', range(5))
problem.addConstraint(lambda a, b: a + b == 5)
problem.addConstraint(lambda a, b: a * b == 6)
solutions = problem.getSolutions()
print solutions
```
-
```
[{'a': 3, 'b': 2}, {'a': 2, 'b': 3}]
```

The End

Thank you!

☻

Any Questions?