MiniZinc, Jupyter and friends...

Sylvain Soliman

Ínria

September 19th, 2018
MiniZinc

Most of the course will use MiniZinc as programming/modelling language

MiniZinc  high-level  \xrightarrow{\text{compile}}  FlatZinc  low-level
MiniZinc syntax

A MiniZinc program may contain parameters

```plaintext
int: i = 1;
int  j;
j = 2;
```

Fixed value (named constants), of type `int`, `float`, `bool` or `string`

May be given in a separate `data` file (`.dzn`)
MiniZinc syntax

A MiniZinc program may contain decision variables

```plaintext
var int: u;
var 1..10: v;
```

Only int or float, given with an optional domain

Parameters and variables can appear in constraints (using the usual arithmetic and Boolean relation operators)

```plaintext
constraint u = 18 * v + 42;
constraint alldifferent([x, y, z]);
```
Additional MiniZinc Syntax

It is possible to define **sets** and **arrays** of objects

```plaintext
set of int: STUDENT = 0..n;
% m[i] is the mark of student number i
array[STUDENT] of var int: m;
```

Iterators over those structures are given

```plaintext
constraint exist(i in STUDENT)
  (m[i] = 20);
constraint forall(i in STUDENT)
  (m[i] <= 20);
```
Final Bits

At least one `solve` statement must appear in a MiniZinc model

```plaintext
solve satisfy;
solve maximize u + 3*v;
solve minimize sum(i in STUDENT)(m[i])
```

`output` takes a list of strings and displays them

Many other things (function or predicate definition, enums, comprehensions, etc.) → in TDs
Jupyter Lab

Work will be done inside Jupyter Lab

New/Next generation of Jupyter Notebooks

Assuming some basic knowledge of Python

Otherwise see: https://docs.python.org/3/tutorial/introduction.html
def df(solutions):
    return pd.concat(
        (pd.DataFrame(sol, index=[idx]) for idx, sol
         in enumerate(solutions)),
        sort=False)

class PicatSatSolver(Solver):
    def __init__(self):
        super().__init__(globals_dir=None)

    def args(self, fzn_file, **kwargs):
        return ['picat', 'fzn_picat_sat', fzn_file]

class OsiCBCSolver(Solver):
    def __init__(self):
        super().__init__(globals_dir=None, support_stats=True, support_mzn=True)

    def args(self, mzn_file, statistics=False, **kwargs):
        if statistics:
            return ['minizinc', '-s', '--solver', 'osicbc', mzn_file]
        return ['minizinc', '--solver', 'osicbc', mzn_file]

sat_solver = PicatSatSolver()
osicbc = OsiCBCSolver()

jovyan@4f68f177b1dc:~$ which --a vim emacs
/usr/bin/vim
/usr/bin/emacs

jovyan@4f68f177b1dc:~$ ls -l --color inf555.py
-rw-rw-r-- 1 jovyan users 957 Aug 30 12:55 inf555.py
jovyan@4f68f177b1dc:~$
```python
import sys
sys.path

['',
 '/home/jovyan',
 '/opt/conda/lib/python36.zip',
 '/opt/conda/lib/python3.6',
 '/opt/conda/lib/python3.6/lib-dynload',
 '/opt/conda/lib/python3.6/site-packages',
 '/opt/conda/lib/python3.6/site-packages/IPython/extensions',
 '/home/jovyan/.ipython']

import os

os.name

'posix'

print('hello world')

def print_hello_world():
    print('hello world')

print_hello_world()
```
Docker

All will be run from **Docker** containers

Same environment for everyone

Do not forget to save your work!
send it by e-mail to Sylvain.Soliman@inria.fr
and Francois.Fages@inria.fr
Setup

1. Download Docker
   https://docs.docker.com/install/

2. Install Docker

3. Pull the image for the course
   
   ```bash
docker pull \
registry.gitlab.inria.fr/soliman/inf555
   ```
Windows Users

You might need Docker Toolbox (and not CE) if you are using Family Edition.

You might need to use 192.168.99.100 instead of localhost for connecting to Jupyter.

If all else fails, use VirtualBox to install an Ubuntu image and follow the instructions to install Docker there.
TD1

Pull the missing files

docker pull \nregistry.gitlab.inria.fr/soliman/inf555/td1

Run on local port 8888 with the work directory of the container pointing to where you launch the command

docker run -p 8888:8888 -v "$PWD":/home/jovyan/work \nregistry.gitlab.inria.fr/soliman/inf555/td1