All you need to know to run your solver in probo, with specific reference to ICCMA’15

Federico Cerutti  Nir Oren  Hannes Straß  Matthias Thimm  Mauro Vallati
Serena Villata

Rev. 4 — 11 November 2014

1 Introduction

This document describes the interfaces that a solver needs to export in order to be executed by probo. For a description of probo and of the ICCMA’15 competition, the reader is invited to study (Cerutti et al., 2014) and to regularly check the official website http://argumentationcompetition.org.

Knowledge of (Cerutti et al., 2014) is a prerequisite for this document.

Remark for the ICCMA’15 Competition
These boxes highlight what is necessary to know for the forthcoming ICCMA’15 competition.

2 Terminology

In this document, we consider the following semantics (Baroni et al., 2011) for Dung’s argumentation framework (Dung, 1995) supported by probo:

- conflict-freeness (hereafter CF);
- admissibility (hereafter ADM);
- complete (hereafter CO);
- grounded (hereafter GR);
- preferred (hereafter PR);
- stable (hereafter ST);
- stage (hereafter STG);
- semi-stable (hereafter SST);
• ideal (hereafter ID);
• CF2 (hereafter CF2).

Semantics considered in ICCMA’15
• complete (hereafter CO);
• grounded (hereafter GR);
• preferred (hereafter PR);
• stable (hereafter ST).

The problems supported by probo discussed in this document are:

• Decision problems:
  1. Credulous acceptance (hereafter DC);
  2. Skeptical acceptance (hereafter DS);

• Enumeration problems:
  1. all the extensions (hereafter EE);
  2. some extension (hereafter SE);
  3. enumerate all the arguments that are credulously inferred (hereafter EC);
  4. enumerate all the arguments that are skeptically inferred (hereafter ES).

Problems considered in ICCMA’15
• Decision problems:
  1. Credulous acceptance (hereafter DC);
  2. Skeptical acceptance (hereafter DS);

• Enumeration problems:
  1. all the extensions (hereafter EE);
  2. some extension (hereafter SE).

probo supports the following three serialisations (see (Cerutti et al., 2014) for examples):

• Aspartix format (hereafter apx);

1For single-status semantics, this problem is equivalent to EE.
• CNF format (hereafter cnf);
• Trivial graph format (hereafter tgf).

File formats considered in ICCMA ’15
• Aspartix format (hereafter apx);
• CNF format (hereafter cnf);
• Trivial graph format (hereafter tgf).

3 Interfaces to probo: Reference Guide

probo interacts with each solver in two ways. First, it queries each solver’s capabilities in terms of supported file types and problems. Secondly, it invokes the solver on a specific argumentation framework for a specific purpose (problem).

Each solver must write — on the standard output — the answers to the invocation. There are three types of answers:

1. YES or NO — for decision problems;
2. [el1, el2, el3] — list of elements (e.g. a list of arguments for credulous/skeptical enumeration problems);
3. [[el1, el2, el3],[...],...] — list of lists of elements (e.g. extensions enumeration);

3.1 Capabilities query

Each solver to be invoked by probo must export the following options for query purposes:

• when invoked without option, the solver must write author and version information to the standard output;

• --formats

when invoked with this parameter the solver must write the list of supported file types to the standard output. Acceptable output are any sublists of:

[apx, cnf, tgf]

• --problems

when invoked with this parameter the solver must write the list of supported problems to the standard output. Acceptable output are any sublists of:

Problems considered in ICCMA ’15
[DC-CO, DC-GR, DC-PR, DC-ST, DS-CO, DS-GR, DS-PR, DS-ST, EE-CO, EE-GR, EE-PR, EE-ST, SE-CO, SE-GR, SE-PR, SE-ST]

For example, if a solver supports credulous acceptance for complete semantics and grounded semantics, and enumeration of stable extensions, the expected output is like
[DC-CO, DC-GR, EE-ST]
(the order in the list does not matter).

3.2 Query Problem’s Answer

Each solver to be invoked by probo must be able to parse and respond to the following options:

- `-f filename` — giving the input file name for a problem;
- `-fo format` — e.g. `-f apx` for specifying that `filename` is in Aspartix format;
- `-p problem` — e.g. `-p DC-PR` for specifying that the problem to be solved is the credulous acceptance w.r.t. preferred semantics;
- `-a additional` — providing additional problem related information. E.g. `-a a1` for specifying that the argument to be checked for credulous acceptance is `a1`.

The syntactically acceptable outputs depend on the type of problem:

- for decision problems, i.e. DC or DS:
  valid output are either YES or NO;
- for enumeration of some extension, enumeration of the arguments that are credulously/skeptically inferred, i.e. SE, EC or ES:
  valid output is a list of arguments (e.g. `[a1, a2]`);
- for extensions enumeration:
  valid output is a list of lists of arguments (e.g. `[[a1,a2],[a3]]`). Please note that this is the case also for single status semantics. In the case no extension exists, the answer must be an empty list `[]`.

4 A tutorial

In this section, we describe, step-by-step, the procedure for allowing probo to invoke a generic solver. For simplicity we consider ArgSemSATv0.2 (http://sourceforge.net/projects/argsemsat/) as the solver.
Since ArgSemSATv0.2 does not export the interfaces requested by probo, we can create a bash script acting as a proxy (Listing 1).

Listing 1: Bash Script for ArgSemSATv0.2

```bash
#!/bin/bash

solver="ArgSemSAT"
fileinput=""
problem=""
format=""

if [ "#$" = "0" ]
then
    echo "ArgSemSATv0.2"
    echo "Federico Cerutti <federico.cerutti@acm.org>"
    echo "Mauro Vallati <mauro.vallati@hud.ac.uk>"
fi

while [ "$1" != "" ]; do
    case $1 in
    "--formats")
        echo '[apx]
        exit 0
        ;;
    "--problems")
        echo '[EE-PR]
        exit 0
        ;;
    "-p")
        shift
        problem=$1
        ;;
    "-f")
        shift
        fileinput=$1
        ;;
    "-fo")
        shift
        format=$1
        ;;
    esac
    shift
    done

if [ "$format" = "apx" -a "$problem" = "EE-PR" ];
then
    res=$(($(dirname $0)/$solver $fileinput -ExtSAT GLUCOSE -sem preferred -df)
    echo -n "[
    echo $res | sed 's/\{|[/g' | sed 's/\}/]/g' | tr -d '\n' | sed 's/}{/},{/g' | sed 's/\|/,,/g' | echo "]"
fi
```

First of all, if the script is invoked without arguments (lines 8–13), then information regarding version and authors is provided.

ArgSemSATv0.2 can work on files in Aspartix format only, therefore the return value of the proxy script when it is invoked with the parameter --formats is [apx] (lines 17–20).

5
Then, since ArgSemSATv0.2 can be tested for performance purposes only on a specific problem, viz. the enumeration of preferred extensions, the return values of the proxy script when it is invoked with the parameter --problems must be [EE-PR] (lines 21–24).

The bash script accepts the parameters -p (lines 24–28), -f (lines 29–32), and -fo (lines 33–36) storing accordingly the provided data.

Finally, if invoked on an EE-PR problem with an apx type file as input, at lines 41–48 the bash script invokes the actual program ArgSemSAT (line 43) and it stores the result of the computation in the variable res. Then, the output is formatted accordingly to probo’s requirements (line 46) by simple string transformation and printed to the standard output.

If you want to test the result, download probo from http://sourceforge.net/projects/probo/, add your solver and your bash script interface to the solvers/ directory, and add your bash script to the array at line 45 of the class net.sf.probo.benchmark.Benchmark, e.g.

```java
private static String[] solvers = {"solvers/tweetysolver-v1.0.6.sh","solvers/argsemsatv0.2.sh"};
```

5 A Configurable Bash Script

As part of the probo distribution, the following bash script is also provided (Listing 2). It can be customised by providing:

1. a information procedure displaying author and version (lines 27–33);
2. a solver procedure which invokes the actual program (lines 36–64);
3. a parse output procedure which parses the output of the actual program in order to adhere to probo’s requests;
4. the list of accepted input type (lines 95–97, please comment the unsupported formats);
5. the list of accepted problems (lines 100–163, please comment the unsupported problems).

This bash script is provided for solvers’ developer convenience only, who are clearly free to implement probo’s interface in other ways.

Listing 2: Generic Bash Script

```bash
#!/bin/bash
# (c)2014 Federico Cerutti <federico.cerutti@acm.org> --- MIT LICENCE
# Generic script interface to probo http://sourceforge.net/projects/probo/
# Please feel free to customize it for your own solver

# function for echoing on standard error
echoerr()
{
    # to remove standard error echoing, please comment the following line
    echo "$@" 1>&2;
}

################################
# C O N F I G U R A T I O N
#
# this script must be customized by defining:
```
1) procedure for printing author and version information of the solver
   (function "information")
2) suitable procedures for invoking your solver (function "solver");
3) suitable procedures for parsing your solver’s output
   (function "parse_output");
4) list of supported format (array "formats");
5) list of supported problems (array "problems").

# output information
function information
{
  # example for ArgSemSATv0.2
  echo "ArgSemSATv0.2"
  echo "Federico Cerutti <federico.cerutti@acm.org>"
  echo "Mauro Vallati <mauro.vallati@hud.ac.uk>"
}

# how to invoke your solver: this function must be customized
function solver
{
  fileinput=$1 # input file with correct path
  format=$2 # format of the input file (see below)
  problem=$3 # problem to solve (see below)
  additional=$4 # additional information, i.e. name of an argument

  # dummy output
  echoerr "input file: $fileinput"
  echoerr "format: $format"
  echoerr "problem: $problem"
  echoerr "additional: $additional"

  # example for ArgSemSATv0.2
  if [ "$format" = "apx" -a "$problem" = "EE-PR" ];
  then
    ./$(dirname $0)/ArgSemSAT $fileinput -ExtSAT GLUCOSE -sem preferred -df
  else
    echoerr "unsupported format or problem"
    exit 1
  fi
}

# how to parse the output of your solver in order to be compliant with probo:
# this function must be customized
# probo accepts solutions of the form:
# [arg1, arg2, ..., argN] for
# 1. some extension enumeration (SE)
# 2. enum. arguments credulously inferred (EC)
# 3. enum. arguments skeptically inferred (ES)
# [[arg1, arg2, ..., argN], [...]...] for extension(s) enumeration
# YES/NO for decision problems
function parse_output
{
  problem=$1
  output="$2"
echoerr "original output: $output"

# example of parsing for ArgSemSATv0.2, which returns "\{arg1, arg2, ...\}\n...
if [ "$problem" = "EE-PR" ];
then
  echo -n "["
  echo $output | sed 's /{/[/g' | sed 's /}/]/g' | tr -d '
' | sed 's /}{/} ,{/g' | sed 's/ /, /g'
  echo "]
else
  echoerr "unsupported format or problem"
  exit 1
fi

# accepted formats: please comment those unsupported
formats[1]="apx" # "aspartix" format
formats[2]="cnf" # conjunctive normal form
formats[3]="tgf" # trivial graph format

# problems that can be accepted: please comment those unsupported
problems[1]="DC-CO" # Decide credulously according to Complete semantics
problems[2]="DC-GR" # Decide credulously according to Grounded semantics
problems[3]="DC-PR" # Decide credulously according to Preferred semantics
problems[4]="DC-ST" # Decide credulously according to Stable semantics
problems[5]="DS-CO" # Decide skeptically according to Complete semantics
problems[6]="DS-GR" # Decide skeptically according to Grounded semantics
problems[7]="DS-PR" # Decide skeptically according to Preferred semantics
problems[8]="DS-ST" # Decide skeptically according to Stable semantics
problems[9]="EE-CO" # Enumerate all the extensions according to Complete semantics
problems[10]="EE-GR" # Enumerate all the extensions according to Grounded semantics
problems[11]="EE-PR" # Enumerate all the extensions according to Preferred semantics
problems[12]="EE-ST" # Enumerate all the extensions according to Stable semantics
problems[13]="SE-CO" # Enumerate some extension according to Complete semantics
problems[14]="SE-GR" # Enumerate some extension according to Grounded semantics
problems[15]="SE-PR" # Enumerate some extension according to Preferred semantics
problems[16]="SE-ST" # Enumerate some extension according to Stable semantics
problems[17]="DC-ADM" # Decide credulously according to admissibility
problems[18]="DC-CF2" # Decide credulously according to CF2 semantics
problems[19]="DC-CF" # Decide credulously according to conflict-freeness
problems[20]="DC-ID" # Decide credulously according to Ideal semantics
problems[21]="DC-SST" # Decide credulously according to Semi-stable semantics
problems[22] = "DC-STG"  # Decide credulously according to Stage semantics
problems[23] = "DS-ADM"  # Decide skeptically according to admissibility
problems[24] = "DS-CF2"  # Decide skeptically according to CF2 semantics
problems[25] = "DS-CF"   # Decide skeptically according to conflict-freeness
problems[26] = "DS-ID"   # Decide skeptically according to Ideal semantics
problems[27] = "DS-SST"  # Decide skeptically according to Semi-stable semantics
problems[28] = "DS-STG"  # Decide skeptically according to Stage semantics
problems[29] = "EC-ADM"  # Enumerate all the arguments credulously inferred according to admissibility
problems[30] = "EC-CF2"  # Enumerate all the arguments credulously inferred according to CF2 semantics
problems[31] = "EC-CF"   # Enumerate all the arguments credulously inferred according to conflict-freeness
problems[32] = "EC-CO"   # Enumerate all the arguments credulously inferred according to Complete semantics
problems[33] = "EC-GR"   # Enumerate all the arguments credulously inferred according to Grounded semantics
problems[34] = "EC-ID"   # Enumerate all the arguments credulously inferred according to Ideal semantics
problems[35] = "EC-PR"   # Enumerate all the arguments credulously inferred according to Preferred semantics
problems[36] = "EC-SST"  # Enumerate all the arguments credulously inferred according to Semi-stable semantics
problems[37] = "EC-STG"  # Enumerate all the arguments credulously inferred according to Stage semantics
problems[38] = "EC-ST"   # Enumerate all the arguments credulously inferred according to Stable semantics
problems[39] = "EE-ADM"  # Enumerate all the extensions according to admissibility
problems[40] = "EE-CF2"  # Enumerate all the extensions according to CF2 semantics
problems[41] = "EE-CF"   # Enumerate all the extensions according to conflict-freeness
problems[42] = "EE-ID"   # Enumerate all the extensions according to Ideal semantics
problems[43] = "EE-SST"  # Enumerate all the extensions according to Semi-stable semantics
problems[44] = "EE-STG"  # Enumerate all the extensions according to Stage semantics
problems[45] = "ES-ADM"  # Enumerate all the arguments skeptically inferred according to admissibility
problems[46] = "ES-CF2"  # Enumerate all the arguments skeptically inferred according to CF2 semantics
problems[47] = "ES-CF"   # Enumerate all the arguments skeptically inferred according to conflict-freeness
problems[48] = "ES-CO"   # Enumerate all the arguments skeptically inferred according to Complete semantics
problems[49] = "ES-GR"   # Enumerate all the arguments skeptically inferred according to Grounded semantics
problems[50] = "ES-ID"   # Enumerate all the arguments skeptically inferred according to Ideal semantics
problems[51] = "ES-PR"   # Enumerate all the arguments skeptically inferred according to Preferred semantics
problems[52] = "ES-SST"  # Enumerate all the arguments skeptically inferred according to Semi-stable semantics
problems[53] = "ES-STG"  # Enumerate all the arguments skeptically inferred according to Stage semantics
problems[54] = "ES-ST"   # Enumerate all the arguments skeptically inferred according to Stable semantics
problems[55]="SE-ADM" # Enumerate some extension according to admissibility
problems[56]="SE-CF2" # Enumerate some extension according to CF2 semantics
problems[57]="SE-CF" # Enumerate some extension according to Conflict-freeness
problems[58]="SE-ID" # Enumerate some extension according to Ideal semantics
problems[59]="SE-SST" # Enumerate some extension according to Semi-stable semantics
problems[60]="SE-STG" # Enumerate some extension according to Stage semantics

# END OF CONFIGURATION SECTION

function list_output
{
    declare -a arr="${!1}"
    check_something_printed=false
    echo -n ']['
    for i in ${arr[@]}; do
        if [ "$check_something_printed" = true ]; then
            echo -n ", "
        fi
        echo -n $i
        check_something_printed=true
        done
    echo ']'
}

function main
{
    if [ "$#" = "0" ]; then
        information
        exit 0
    fi
    local local_problem=""
    local local_fileinput=""
    local local_format=""
    local local_additional=""
    while [ "$1" != "" ]; do
        case $1 in
            "--formats")
                list_output formats[@]
                exit 0
                ;;
            "--problems")
                list_output problems[@]
                exit 0
                ;;
            "-p")
                shift
                local_problem=$1
                ;;
            "-f")
                shift
                local_fileinput=$1
                ;;
            "-e")
                shift
                local_environment=$1
                ;;
            "-c")
                shift
                local_configuration=$1
                ;;
            "-s")
                shift
                local_skel=$1
                ;;
            "--config")
                shift
                local_config=$1
                ;;
            "--execute")
                shift
                local_execute=$1
                ;;
            "--script")
                shift
                local_script=$1
                ;;
            "--test")
                shift
                local_test=$1
                ;;
            "--trace")
                shift
                local_trace=$1
                ;;
            "--debug")
                shift
                local_debug=$1
                ;;
            "--version")
                shift
                local_version=$1
                ;;
            "--help")
                shift
                local_help=$1
                ;;
            *)
                echo "unknown option: $1"
                exit 1
                ;;
        esac
    fi
}

shift local_format=$1
 ;;
shift local_additional=$1
 ;;
esac
shift
done
res=$(solver $local_fileinput $local_format $local_problem $local_additional)
parse_output $local_problem "$res"
}
main $@
exit 0

References

