Interactive debugging of non-ground ASP programs

Carmine Dodaro[1] Philip Gasteiger[2] Benjamin Musitsch[2]
Francesco Ricca[1] Kostyantyn Shchekotykhin[2]

[1]University of Calabria, Italy
[2]Alpen-Adria-Universität Klagenfurt, Austria

Lexington, Kentucky
LPNMR 2015

Outline

Outline

Context

Answer Set Programming (ASP)

- declarative programming paradigm
- strong theoretical basis
- availability of efficient implementations
- ease in representing complex problems

Idea

1. logic programs represent computational problems
2. answer sets correspond to solutions

Context

Answer Set Programming (ASP)

- declarative programming paradigm
- strong theoretical basis
- availability of efficient implementations
- ease in representing complex problems

Idea

1. logic programs represent computational problems
2. answer sets correspond to solutions

...and then the solution is not correct!

ASP encoding: graph coloring

Goal

Input: A direct graph $G =< V, E >$ and a set of three colors
Output: A color assignment for each node in V

3-Graph Coloring Problem

% Compute nodes from arcs
$node(X) \leftarrow arc(X, Y)$
$node(X) \leftarrow arc(Y, X)$

% Assign a color to each node
$col(X, blue) \mid col(X, red) \mid col(X, yellow) \leftarrow node(X)$

% Different colors for adjacent nodes
$\leftarrow col(X, C1), col(Y, C2), arc(X, Y)$

ASP encoding: graph coloring

Goal

Input: A direct graph $G =< V, E >$ and a set of three colors
Output: A color assignment for each node in V

3-Graph Coloring Problem

% Compute nodes from arcs
$node(X) \leftarrow arc(X, Y)$
$node(X) \leftarrow arc(Y, X)$

% Assign a color to each node
$col(X, blue) \mid col(X, red) \mid col(X, yellow) \leftarrow node(X)$

% Different colors for adjacent nodes
$\leftarrow col(X, C1), col(Y, C2), arc(X, Y), C1 = C2$

Motivation

- ASP encodings are usually compact compared to C++ programs
  - the encoding of Valves Location Problem from the 5th competition is composed by ~100 lines of code
  - the file *clasp_options.cpp* contains ~1000 lines of code
- However, faulty detection of ASP programs may be tedious
  - finding errors in (even small) ASP programs requires a lot of time
  - debuggers make the development process faster and more comfortable

Existing debuggers

- Algorithmic/native approaches
  - DLV debugger, IDEAS, stepping framework
- Declarative approaches
  - SPOCK, OUROBOROS
  - ASP to debug ASP
  - represents the input program in a reified form
- Limitations
  - some of them work only for ground programs
  - declarative approaches cause a blow up in the size of the grounded program
  - a novice might find it difficult to understand the output of debuggers

Contribution

- A new debugging technique
  - works on non-ground ASP programs
  - no grounding blow up
  - the output is the faulty rule(s)
- Implementation of the technique in DWASP

Outline

Introduction

- A bug in an ASP program is revealed when
  1. one or more answer sets are incorrect
  2. one or more answer sets are missing
- The definition of test cases
  - is a good practice of software engineering
  - two meanings: unit testing [De Vos et al., TPLP 2012; Febbraro et al., INAP/WLP 2011] and coverage testing [Janhunen et al., ECAI 2010; Janhunen et al., LPNMR 2011]
  - is supported by modern tools like ASPIDE and SEALION

Introduction

- A bug in an ASP program is revealed when
  1. one or more answer sets are incorrect
  2. one or more answer sets are missing
- The definition of test cases
  - is a good practice of software engineering
  - two meanings: unit testing [De Vos et al., TPLP 2012; Febbraro et al., INAP/WLP 2011] and coverage testing [Janhunen et al., ECAI 2010; Janhunen et al., LPNMR 2011]
  - is supported by modern tools like ASPIDE and SEALION
- In the following the faulty ASP program is assumed to be *Incoherent* in presence of one or more test cases

Debug of an incoherent ASP program

- **Input:** an incoherent program Π
- **Output:** the faulty rule(s) causing the problem
- **Debugging:** based on the concept of unsatisfiable core
  - a set of rules causing the incoherence of Π
  - it is computed by modern ASP solvers during the solving process

Unsatisfiable core

Input program

$r_1: \quad a \mid b \leftarrow$ $\qquad r_2: \quad c \leftarrow a$ $\qquad r_3: \quad c \leftarrow b$

$r_4: \quad d \mid e \leftarrow$ $\qquad r_5: \quad e \leftarrow a$ $\qquad r_6: \quad d \leftarrow b$

Test case

Execution

The program is coherent: $\{a, c, e\}$ is an answer set

Unsatisfiable core

Input program

$r_1 : \quad a \mid b \leftarrow$ $\qquad r_2 : \quad c \leftarrow a$ $\qquad r_3 : \quad c \leftarrow b$

$r_4 : \quad d \mid e \leftarrow$ $\qquad r_5 : \quad e \leftarrow a$ $\qquad r_6 : \quad d \leftarrow b$

$r_7 : \quad \leftarrow c$

Test case

FALSE(C).

Execution

The program is now incoherent

Unsatisfiable core

Input program

$r_1 : \quad a \mid b \leftarrow$ $\qquad r_2 : \quad c \leftarrow a$ $\qquad r_3 : \quad c \leftarrow b$

$r_4 : \quad d \mid e \leftarrow$ $\qquad r_5 : \quad e \leftarrow a$ $\qquad r_6 : \quad d \leftarrow b$

$r_7 : \quad \leftarrow c$

Test case

FALSE(C).

Execution

The program is now incoherent: $\{r_1, r_2, r_3, r_7\}$ is an unsatisfiable core

DWASP: determine query (1)

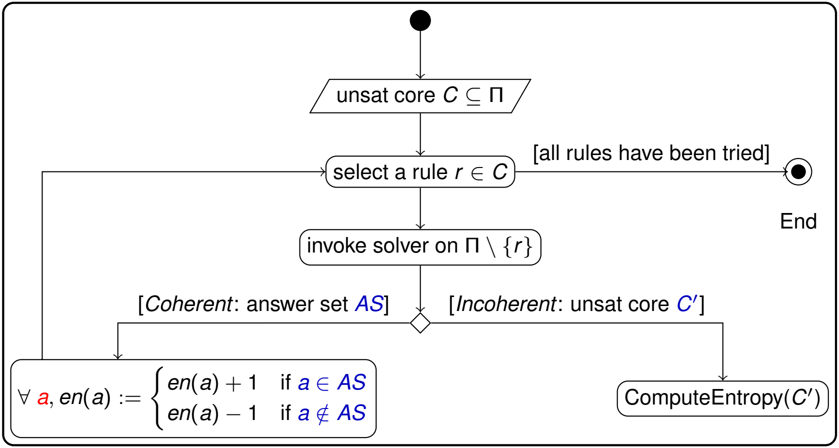
- An unsatisfiable core might contain a high number of rules
  - minimize the core (e.g. using QUICKXPLAIN)
  - the core might still be huge!

    $\rightarrow$ is not informative in this case

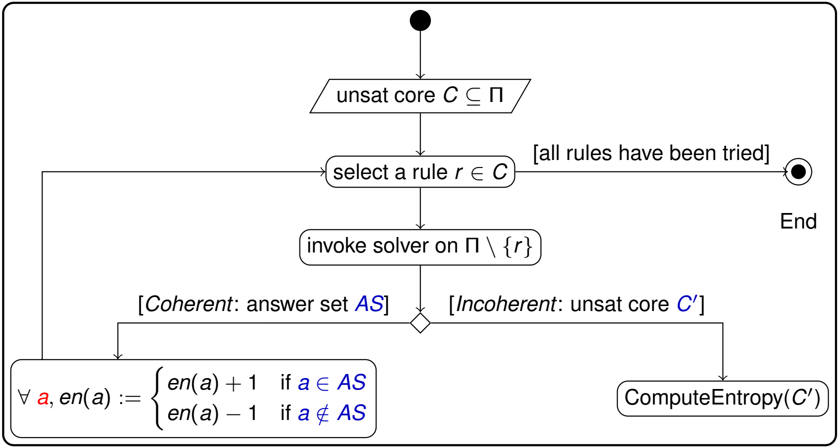
DWASP: determine query (1)

- An unsatisfiable core might contain a high number of rules
  - minimize the core (e.g. using QUICKXPLAIN)
  - the core might still be huge!

    $\rightarrow$ is not informative in this case
- A query-based approach to obtain smaller cores
  - ask the user whether an atom must be true or false
  - too many queries may be tedious
  - maximize the SPLIT-IN-HALF measure [Shchekotykhin and Friedrich, ISWC 2010]



ComputeEntropy



ComputeEntropy

The query atom q is the one whose $en(q)$ is the closest to 0

Input program

$r_1: \quad a \mid b \leftarrow$ $\qquad r_2: \quad c \leftarrow a$ $\qquad r_3: \quad c \leftarrow b$

$r_4: \quad d \mid e \leftarrow$ $\qquad r_5: \quad e \leftarrow a$ $\qquad r_6: \quad d \leftarrow b$

$r_7: \quad \leftarrow c$

Unsat core

$\{r_1, r_2, r_3, r_7\}$

Execution

Entropy

$en(a) := 0$

$en(b) := 0$

$en(c) := 0$

$en(d) := 0$

$en(e) := 0$

Input program

$r_2 : \quad c \leftarrow a$ $\qquad r_3 : \quad c \leftarrow b$

$r_4 : \quad d \mid e \leftarrow$ $\qquad r_5 : \quad e \leftarrow a$ $\qquad r_6 : \quad d \leftarrow b$

$r_7 : \quad \leftarrow c$

Unsat core

$\{r_1, r_2, r_3, r_7\}$

Execution

$\Pi \setminus r_1 : \{e\}$

Entropy

$en(a) :=$ -1

$en(b) :=$ -1

$en(c) :=$ -1

$en(d) :=$ -1

$en(e) := 1$

Input program

$r_1: \quad a \mid b \leftarrow$ $\qquad r_2: \quad c \leftarrow a$ $\qquad r_3: \quad c \leftarrow b$

$r_4: \quad d \mid e \leftarrow$ $\qquad r_5: \quad e \leftarrow a$ $\qquad r_6: \quad d \leftarrow b$

$r_7: \quad \leftarrow c$

Unsat core

$\{r_1, r_2, r_3, r_7\}$

Execution

$\Pi \setminus r_1 : \{e\}$

$\Pi \setminus r_2 : \{a, e\}$

$\Pi \setminus r_3 : \{b, d\}$

$\Pi \setminus r_7 : \{a, c, e\}$

Entropy

$en(a) := 0$

$en(b) :=$ -2

$en(c) :=$ -2

$en(d) :=$ -2

$en(e) := 2$

Input program

$r_1: \quad a \mid b \leftarrow$ $\qquad r_2: \quad c \leftarrow a$ $\qquad r_3: \quad c \leftarrow b$

$r_4: \quad d \mid e \leftarrow$ $\qquad r_5: \quad e \leftarrow a$ $\qquad r_6: \quad d \leftarrow b$

$r_7: \quad \leftarrow c$

Unsat core

$\{r_1, r_2, r_3, r_7\}$

Execution

$\Pi \setminus r_1 : \{e\}$

$\Pi \setminus r_2 : \{a, e\}$

$\Pi \setminus r_3 : \{b, d\}$

$\Pi \setminus r_7 : \{a, c, e\}$

Entropy

$en(a) := 0$

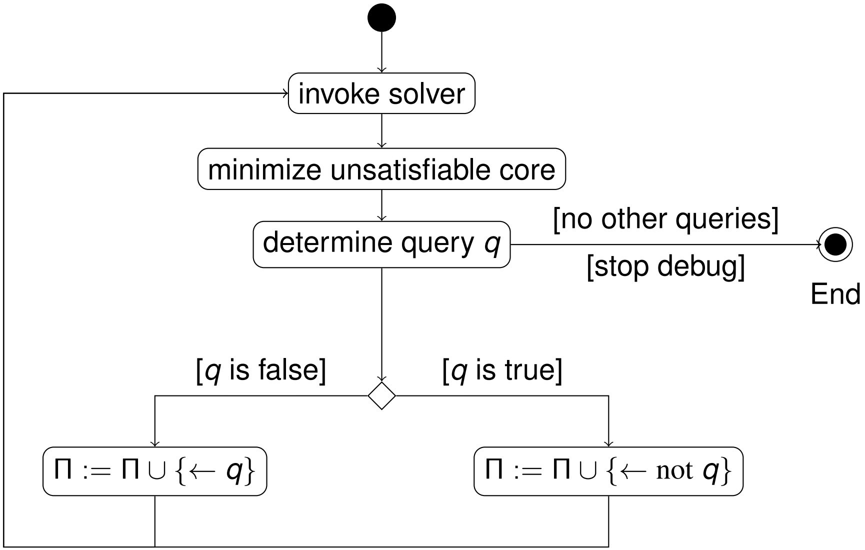
$en(b) :=$ -2

$en(c) :=$ -2

$en(d) :=$ -2

$en(e) := 2$

DWASP: debugging session



Implementation details

- GRINGO-WRAPPER
  - disables the simplifications of GRINGO
  - wrong rules can lead to unintended simplifications
  - adorns the program to label the rules
- WASP is used as internal solvers
  - used as black box exploiting its incremental interface
  - other ASP solvers might be used

Comparison of the grounding size (1)

Instance	GRINGO	GRINGO-WRAPPER	OUROBOROS
GraphCol1-125	6 145	8 031	19 020
GraphCol11-130	6 455	8 416	19 845
GraphCol21-135	7 269	9 305	21 174
GraphCol30-135	6 597	8 633	20 502
GraphCol31-140	7 467	9 578	21 887
GraphCol40-140	8 097	10 208	22 517
GraphCol41-145	8 260	10 446	23 195
GraphCol51-120	8 773	11 034	24 223
Hanoi09-28	31 748	94 166	1 739 800
Hanoi11-30	34 056	100 942	1 864 222
Hanoi15-34	38 672	114 524	2 112 986
Hanoi16-40	27 137	80 615	1 491 281
Hanoi22-60	28 311	84 644	1 678 483
Hanoi38-80	34 044	100 942	1 864 250
Hanoi41-100	31 738	94 166	1 739 830
Hanoi47-120	25 968	77 227	1 429 695

Comparison of the grounding size (2)

Instance	GRINGO	GRINGO-WRAPPER	OUROBOROS
KnightsTour01-8	1 384	3 413	12 985 716
KnightsTour03-12	3 356	8 652	>72 244 034
KnightsTour05-16	6 192	16 285	>69 494 641
KnightsTour06-20	9 892	26 321	>62 785 993
KnightsTour07-30	22 922	61 911	>59 166 564
KnightsTour08-40	41 352	112 501	>54 944 042
KnightsTour09-46	55 002	150 055	>56 443 633
KnightsTour10-50	65 182	178 094	>62 402 315
PartnerUnits176-24	12 563	14 218	102 023
PartnerUnits23-30	39 231	42 106	276 645
PartnerUnits29-40	59 979	64 413	629 639
PartnerUnits207-58	158 564	168 289	2 726 182
PartnerUnits204-67	218 808	231 083	4 280 282
PartnerUnits175-75	682 015	699 472	8 604 415
PartnerUnits52-100	952 363	979 603	20 125 857
PartnerUnits115-100	952 369	979 759	20 317 011

Outline

Conclusion

Contribution

1. a new debugging technique for non-ground ASP programs
2. no blow up in the size of the grounding
3. the output of the debugger is the faulty rule(s)

Current status

- implementation of DWASP-GUI
- integration of DWASP with ASPIDE

Future work

- annotate the “trusty” rules
- support weak constraints
- integrate DWASP with SEALION
- disable the simplifications of GRINGO
- hybrid approach

Tools

1. DWASP: `https://github.com/gaste/dwasp`
2. GRINGO-WRAPPER: `https://github.com/gaste/gringo-wrapper`
3. DWASP-GUI (BETA): `https://github.com/gaste/dwasp-gui`
4. ASPIDE: `http://www.mat.unical.it/ricca/aspide`
5. GRINGO: `http://sourceforge.net/projects/potassco/files/gringo/`
6. WASP: `http://alviano.github.io/wasp/`

Tools

1. DWASP: https://github.com/gaste/dwasp
2. GRINGO-WRAPPER: https://github.com/gaste/gringo-wrapper
3. DWASP-GUI (BETA): https://github.com/gaste/dwasp-gui
4. ASPIDE: http://www.mat.unical.it/ricca/aspide
5. GRINGO: http://sourceforge.net/projects/potassco/files/gringo/
6. WASP: http://alviano.github.io/wasp/

Thank you!