Reasoning About Strong Inconsistency in ASP

Carlos Mencía[1] and Joao Marques-Silva[2]

[1]University of Oviedo, Spain

[2]ANITI, University of Toulouse, France

SAT 2020

Overview

Analysis of unsatisfiable formulas

- Core tasks: minimal explanations (MUS) and corrections (MCS)
- Software fault localization, spreadsheet debugging, type error debugging, axiom pinpointing, etc.

Non-monotonic logics

- Non-monotonicity raises several difficulties
- Recent notion: **strong inconsistency**

Contribution

- Algorithms for reasoning about inconsistency in SAT are applicable to reasoning about strong inconsistency in non-monotonic logics
- We focus on Answer Set Programming (ASP)

Outline

1. Preliminaries
2. Reasoning about Strongly Inconsistent ASP Programs
3. Preliminary Results
4. Conclusions

Preliminaries: SAT

CNF formula $\mathcal{F}$ over set of variables $V(\mathcal{F}) = \{x_1, ..., x_n\}$

- Conjunction of clauses, where a clause is a disjunction of literals and a literal is a variable x or its negation $\neg x$
- Also: set of clauses

Interpretation μ: $V(\mathcal{F}) \to \{0, 1\}$

- If μ satisfies $\mathcal{F}$, it is called a model
- Maximal model: irreducible set of variables assigned value 0

Preliminaries: SAT

Given an unsatisfiable CNF formula $\mathcal{F}$

- Minimal Unsatisfiable Subset (MUS): irreducible unsatisfiable subset of $\mathcal{F}$
- Minimal Correction Subset (MCS): irreducible subset of $\mathcal{F}$ whose removal renders satisfiability

Hitting Set Duality

- MUSes are **minimal hitting sets** of MCSes and vice versa

Preliminaries: MSMP

Minimal Sets over a Monotone Predicate (MSMP) [MSJB13, MSJM17]

- Predicate $p : 2^{\mathcal{R}} \rightarrow \{0, 1\}$, with $\mathcal{R}$ reference set
- Monotone: If $p(\mathcal{R}_0)$, then $p(\mathcal{R}_1)$ for all $\mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \mathcal{R}$
- Mimimal set: $p(\mathcal{M})$ and for all $\mathcal{M}' \subsetneq \mathcal{M}$, $\neg p(\mathcal{M}')$

Examples:

- MUSes: $p(\mathcal{W}) \triangleq \neg\mathsf{SAT}(\mathcal{W})$, with $\mathcal{W} \subseteq \mathcal{R}$ and $\mathcal{R} \triangleq \mathcal{F}$
- MCSes: $p(\mathcal{W}) \triangleq \mathsf{SAT}(\mathcal{R} \setminus \mathcal{W})$, with $\mathcal{W} \subseteq \mathcal{R}$ and $\mathcal{R} \triangleq \mathcal{F}$
- **More examples**: prime implicants/implicates, minimal/maximal models, autarkies, backbone literals, ...

Preliminaries: Answer Set Programming

A logic program is a finite set of rules $P = \{r_1, ..., r_k\}$

- Rule r: $a \leftarrow b_1, ..., b_m, \text{not } c_{m+1}, ..., \text{not } c_n.$
  - $\text{head}(r) = a$
  - $\text{body}(r) = b_1, ..., b_m, \text{not } c_{m+1}, ..., \text{not } c_n$
- Fact: rule with empty body
- Choice rule: $n \leq \{a_1, ..., a_k\}$

Semantics

- Reduct of P w.r.t. set I of atoms:

$$P^I = \{a \leftarrow b_1, ..., b_m \mid r \in P, I \cap \{c_{m+1}, ..., c_n\} = \emptyset\}$$

- The set I is an **answer set** of P if it is a minimal model of P^I
- P is consistent iff it has at least one answer set

Preliminaries: Answer Set Programming

P :

$p \leftarrow \text{not } q.$

Answer set: $\{p\}$

Preliminaries: Answer Set Programming

P :

$p \leftarrow \text{not } q.$

$\leftarrow p.$

Inconsistent

Preliminaries: Answer Set Programming

P :

$$p \leftarrow \text{not } q.$$
$$\leftarrow p.$$
$$q \leftarrow \text{not } r.$$

Answer set: $\{q\}$

Preliminaries: Answer Set Programming

P :

$p \leftarrow \text{not } q.$

$\leftarrow p.$

$q \leftarrow \text{not } r.$

$r.$

Inconsistent

Preliminaries: Answer Set Programming

P :

$p \leftarrow \text{not } q.$

$\leftarrow p.$

$q \leftarrow \text{not } r.$

$r.$

$\leftarrow r.$

Inconsistent

Preliminaries: Answer Set Programming

$P = B \cup S$

- B is background knowledge, assumed consistent
- S consists of rules that can be dropped

Strong Inconsistency

[BTU17; BTU19]

- $P' = B \cup S'$ (with $S' \subseteq S$) is strongly P-inconsistent iff $P'' = B \cup S''$ is inconsistent for all $S' \subseteq S'' \subseteq S$

Minimal explanations and corrections

- Minimal Strongly P-inconsistent Subset (MSIS): irreducible $M \subseteq S$ s.t. $B \cup M$ is strongly P-inconsistent
- Minimal Strong P-inconsistency Correction Subset (MSICS): irreducible $C \subseteq S$ s.t. $B \cup (S \setminus C)$ is not strongly P-inconsistent

Preliminaries: Answer Set Programming

Example: $P_{ex} = B_{ex} \cup S_{ex}$, with $B_{ex} = \emptyset$ and $S_{ex} = \{r_1, r_2, r_3, r_4, r_5\}$

r_1	r_2	r_3	r_4	r_5
$p \leftarrow \text{not } q.$	$\leftarrow p.$	$q \leftarrow \text{not } r.$	$r.$	$\leftarrow r.$

Preliminaries: Answer Set Programming

Example: $P_{ex} = B_{ex} \cup S_{ex}$, with $B_{ex} = \emptyset$ and $S_{ex} = \{r_1, r_2, r_3, r_4, r_5\}$

$$\underset{r_1}{p \leftarrow \text{not } q.} \quad \underset{r_2}{\leftarrow p.} \quad \underset{r_3}{q \leftarrow \text{not } r.} \quad \underset{r_4}{r.} \quad \underset{r_5}{\leftarrow r.}$$

MSISes:
$M_1 = \{r_1, r_2, r_4\}$
$M_2 = \{r_4, r_5\}$

MSICSes:
$C_1 = \{r_1, r_5\}$
$C_2 = \{r_2, r_5\}$
$C_3 = \{r_4\}$

Preliminaries: Answer Set Programming

Example: $P_{ex} = B_{ex} \cup S_{ex}$, with $B_{ex} = \emptyset$ and $S_{ex} = \{r_1, r_2, r_3, r_4, r_5\}$

r_1	r_2	r_3	r_4	r_5
$p \leftarrow \text{not } q.$	$\leftarrow p.$	$q \leftarrow \text{not } r.$	$r.$	$\leftarrow r.$

MSISes:
$M_1 = \{r_1, r_2, r_4\}$
$M_2 = \{r_4, r_5\}$

MSICSes:
$C_1 = \{r_1, r_5\}$
$C_2 = \{r_2, r_5\}$
$C_3 = \{r_4\}$

- $\{r_1, r_2\}$ is inconsistent, but not strongly P_{ex}-inconsistent (e.g. $\{r_1, r_2, r_3\}$ is consistent)

Preliminaries: Answer Set Programming

Example: $P_{ex} = B_{ex} \cup S_{ex}$, with $B_{ex} = \emptyset$ and $S_{ex} = \{r_1, r_2, r_3, r_4, r_5\}$

r_1	r_2	r_3	r_4	r_5
$p \leftarrow \text{not } q.$	$\leftarrow p.$	$q \leftarrow \text{not } r.$	$r.$	$\leftarrow r.$

MSISes:
$M_1 = \{r_1, r_2, r_4\}$
$M_2 = \{r_4, r_5\}$

MSICSes:
$C_1 = \{r_1, r_5\}$
$C_2 = \{r_2, r_5\}$
$C_3 = \{r_4\}$

Preliminaries

Goal: Extraction and enumeration of MSISes and MSICSes

Related work

- Debugging ASP programs: `spock`, `Ouroboros`, `DWASP`, ...
- SAT algorithms for maximal consistent subsets in ASP [JMS17]
- Set enumeration approach for computing MSISes [BTU19]

Reasoning About Strongly Inconsistent ASP Programs

Monotonicity property

- All supersets (up to P) of a strongly P-inconsistent subprogram are strongly P-inconsistent too

Predicate $\mathrm{SAT}^{+}(B, S, R)$, with $P = B \cup S$ and $R \subseteq S$

- Is there S', with $R \subseteq S' \subseteq S$ s.t. the program $B \cup S'$ is consistent?
- $\mathrm{SAT}^{+}(B, S, R)$ is false iff program $B \cup R$ is strongly P-inconsistent

Reductions to MSMP

- MSIS: $p(\mathcal{W}) \triangleq \neg\mathrm{SAT}^{+}(B, S, \mathcal{W})$, with $\mathcal{W} \subseteq \mathcal{R}$ and $\mathcal{R} \triangleq S$
- MSICS: $p(\mathcal{W}) \triangleq \mathrm{SAT}^{+}(B, S, S \setminus \mathcal{W})$, with $\mathcal{W} \subseteq \mathcal{R}$ and $\mathcal{R} \triangleq S$

Reasoning About Strongly Inconsistent ASP Programs

Computing a minimal set

- Algorithms for MSMP: Deletion, QuickXplain, Progression, ...
- Iteratively invoke oracle for $\mathrm{SAT}^{+}(B, S, R)$

Deletion-based approach

1. Initialize $\mathcal{M} = \mathcal{R}$
2. For each $u \in \mathcal{M}$
   - If $p(\mathcal{M} - \{u\})$, drop u from $\mathcal{M}$
   - Otherwise, keep u in $\mathcal{M}$
3. Return $\mathcal{M}$

Reasoning About Strongly Inconsistent ASP Programs

Computing a minimal set

- Algorithms for MSMP: Deletion, QuickXplain, Progression, ...
- Iteratively invoke oracle for $\mathrm{SAT}^{+}(B, S, R)$

Deletion-based approach: MSIS ($\mathcal{R} \triangleq S$)

1. Initialize $\mathcal{M} = \mathcal{R}$
2. For each $u \in \mathcal{M}$
   - If $\neg\mathrm{SAT}^{+}(B, S, \mathcal{M} - \{u\})$ holds, drop u from $\mathcal{M}$
   - Otherwise, keep u in $\mathcal{M}$
3. Return $\mathcal{M}$

Reasoning About Strongly Inconsistent ASP Programs

Computing a minimal set

- Algorithms for MSMP: Deletion, QuickXplain, Progression, ...
- Iteratively invoke oracle for $\mathrm{SAT}^{+}(B, S, R)$

Deletion-based approach: MSICS ($\mathcal{R} \triangleq S$)

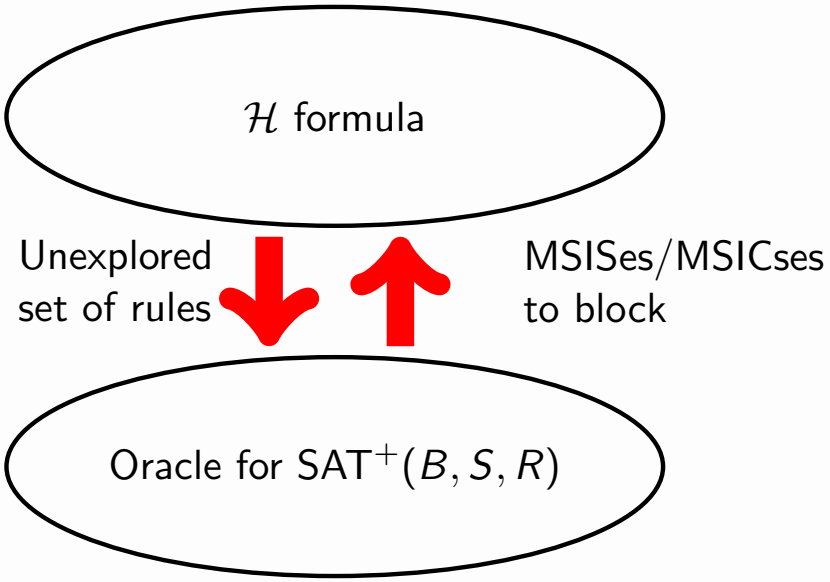
1. Initialize $\mathcal{M} = \mathcal{R}$
2. For each $u \in \mathcal{M}$
   - If $\mathrm{SAT}^{+}(B, S, S \setminus (\mathcal{M} - \{u\}))$ holds, drop u from $\mathcal{M}$
   - Otherwise, keep u in $\mathcal{M}$
3. Return $\mathcal{M}$

Optimization: Exploit witness after positive answers

Reasoning About Strongly Inconsistent ASP Programs

Enumerating minimal sets

- MARCO: exploits hitting set duality relationship [LPMMS16]



While SAT($\mathcal{H}$)

1. Compute **maximal model** of $\mathcal{H}$, inducing the set of rules $R \subseteq S$
2. If $\neg$SAT$^+(B, S, R)$ reduce R to an MSIS and block it in $\mathcal{H}$
3. Else, $(S \setminus R)$ is an MSICS; block it in $\mathcal{H}$

Two variants

- `eMax` (maximal models): preference to MSISes
- `eMin` (minimal models): preference to MSICSes

Reasoning About Strongly Inconsistent ASP Programs

Implementing $\mathrm{SAT}^{+}(B, S, R)$

- Modified program with *selector* atoms and choice rules [JMS17]
- For a set of atoms $\mathcal{A}$, choice($\mathcal{A}$) denotes the rule $0 \leq \{a_1, ..., a_k\}$

Procedure (given $P = B \cup S$)

1. Create a fresh atom s_i for each rule $r_i \in S$
2. Define $P_s = B \cup S_s$
   - For each rule $r_i \in S$, S_s includes the rule $\mathrm{head}(r_i) \leftarrow \mathrm{body}(r_i), s_i$.
3. For testing $\mathrm{SAT}^{+}(B, S, R)$, invoke ASP solver on the program:

$$P' = P_s \cup \cup_{s \in s(R)} \{s\} \cup \mathrm{choice}(s(S \setminus R))$$

where $s(R)$ denotes the selector atoms for the rules in R

Preliminary Results

Prototype

- Written in Python 2.7, interfacing the ASP solver `clingo`
- Maximal and minimal models computed using the tool mcsls

Benchmarks (360 instances in all)

- Domains: *Graceful graphs*, *Knight Tour with Holes* and *Solitaire*
- Random inconsistent programs $P = B \cup S$, where B encodes the problem domain and S contains facts specific to each instance

Experiments

- Evaluate extraction and enumeration of MSISes and MSICSes
- Limits: 3600 seconds, 4 GB memory

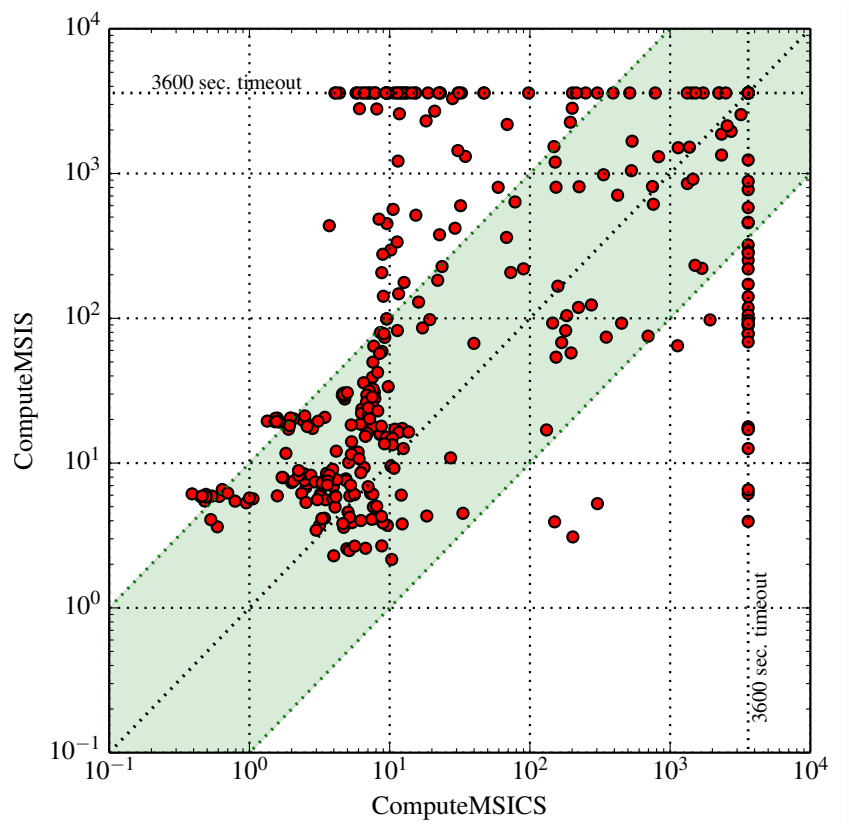
Preliminary Results

Extracting a single minimal set

- MSICSes computed faster than MSISes

Number of solved instances

	# solved
ComputeMSIS	295
ComputeMSICS	**317**



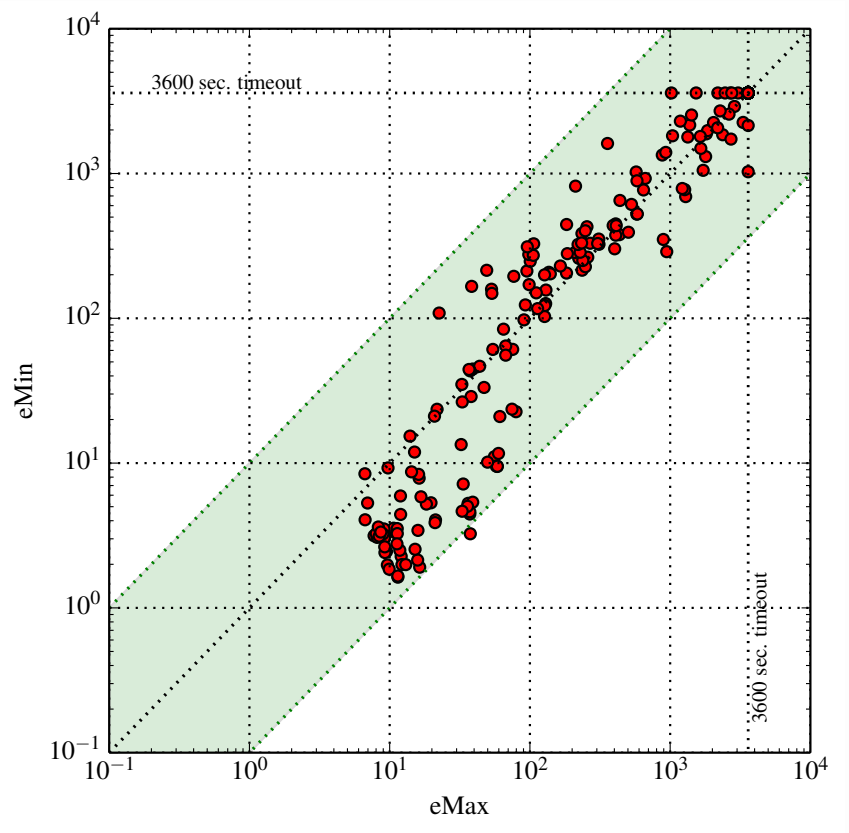
Preliminary Results

Enumerating minimal sets

- No clear winner in terms of running times

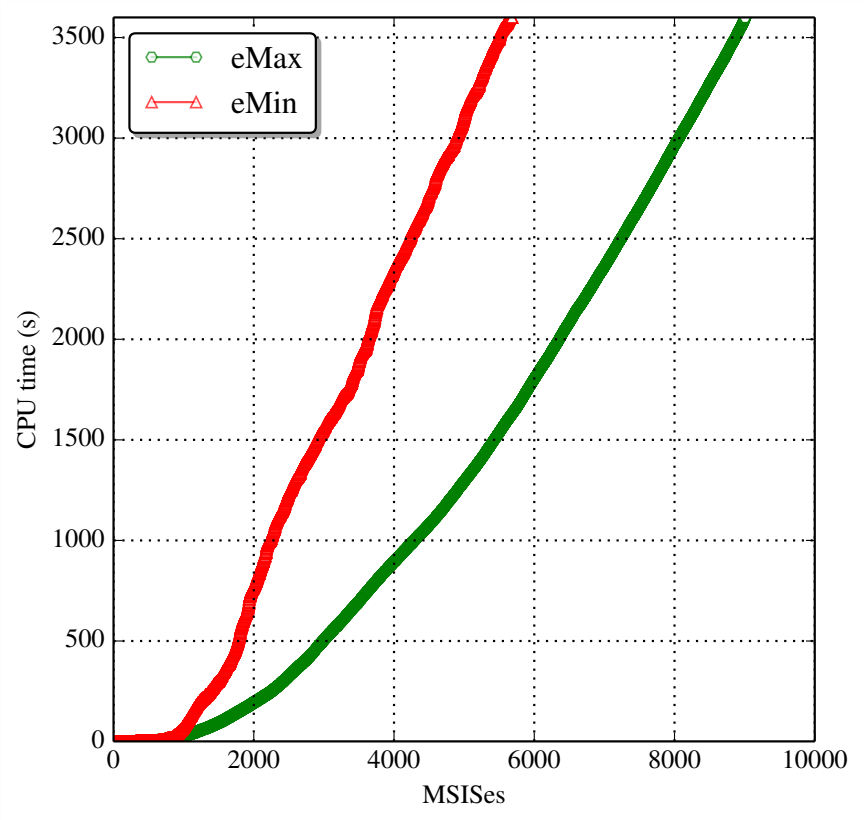
Number of solved instances

	# solved
eMax	**172**
eMin	167

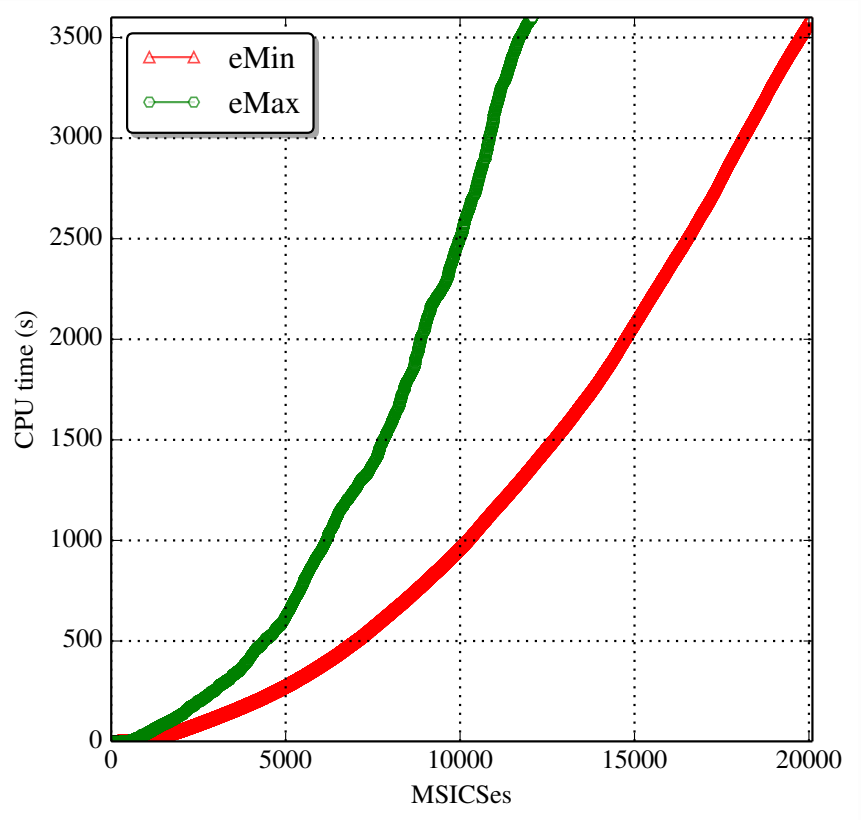


Preliminary Results

Number of reported sets



Reported MSISes



Reported MSICSes

Conclusions

Strong Inconsistency

- Reasoning about inconsistency in non-monotonic logics
- Minimal explanations (MSISes) and corrections (MSICSes)

MSISes and MSICSes are minimal sets over a monotone predicate

- Large body of algorithms readily available
- Encouraging results in ASP

Thank You!