26th International Joint Conference on Artificial Intelligence (IJCAI-17), Melbourne, Australia, August 19–25, 2017

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Knowledge-Based Systems Group

Conflict-driven ASP Solving with External Sources and Program Splits

Christoph Redl

redl@tuwien.ac.at

1. Motivation

HEX-programs extend ASP by **external sources**:

- Rule bodies may contain external atoms of the form
$$\&p[q_1, \ldots, q_k](t_1, \ldots, t_l), \quad \text{where}$$
p ... external predicate name, q_i ... input terms, t_j ... output terms.

Semantics:
$1 + k + l$-ary Boolean oracle function $f_{\&p}$:
ground external atom $\&p[q_1, \ldots, q_k](t_1, \ldots, t_l)$ is true under assignment A iff $f_{\&p}(A, q_1, \ldots, q_k, t_1, \ldots, t_l) = \mathbf{T}$.

Example: Set Partitioning

$$P = \left\{ \begin{array}{l} d(a_1). \ldots . d(a_n). \\ r_1 : p(X) \leftarrow d(X), \&diff[d, q](X). \\ r_2 : q(X) \leftarrow d(X), \&diff[d, p](X). \end{array} \right\}$$

Problem:

- Due to value invention, grounding nonmonotonic external atoms is expensive.
- Previous remedy: split program into components (see Block 4.).
- But: deteriorates conflict-driven solving techniques.

Our Solution:

- Keep splits, but compute reasons for inconsistent components.
- Propagate them as constraints to predecessor components.

2. Inconsistency Reasons (IRs)

Characterization of Program Inconsistency [Redl, 2017]:
Let P be a HEX-program and D be a domain of atoms.

An inconsistency reason (IR) of P wrt. D is a pair $R = (R^+, R^-)$ of sets of atoms $R^+ \subseteq D$ and $R^- \subseteq D$ with $R^+ \cap R^- = \emptyset$ s.t. $P \cup facts(I)$ is inconsistent for all $I \subseteq D$ with $R^+ \subseteq I$ and $R^- \cap I = \emptyset$.

Example

Consider $P = \{\leftarrow a, \mathbf{not}\, c;\ d \leftarrow b.\}$ and $D = \{a, b, c\}$.

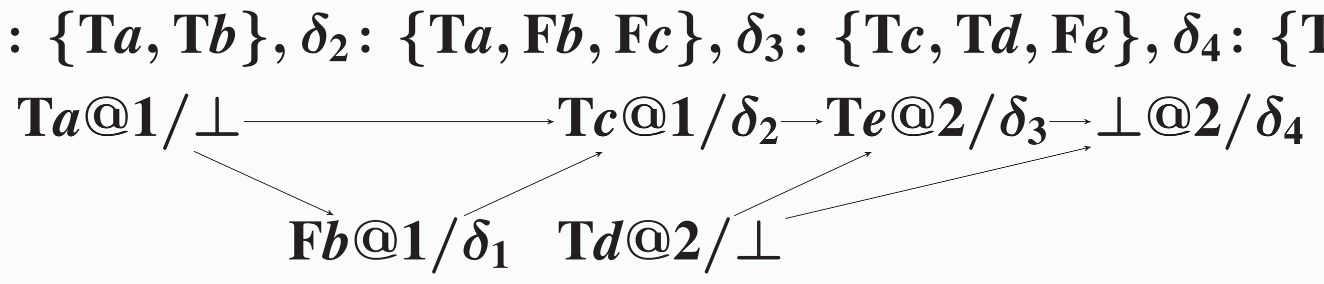
An IR is $R = (\{a\}, \{c\})$ because $P \cup facts(I)$ is inconsistent for all $I \subseteq D$ such that $a \in I$ and $c \notin I$.

3. Computing Inconsistency Reasons

Ground Case:
Consider the implication graph: its nodes are literals in the current assignment, edges represent implications; nodes $\perp$ represent conflicts.

Example

$$\Delta = \{\delta_1 : \{\mathbf{T}a, \mathbf{T}b\}, \delta_2 : \{\mathbf{T}a, \mathbf{F}b, \mathbf{F}c\}, \delta_3 : \{\mathbf{T}c, \mathbf{T}d, \mathbf{F}e\}, \delta_4 : \{\mathbf{T}d, \mathbf{T}e\}\}.$$



- To find a reason for a literal l being true or a conflict $\perp$ in terms of a set D, find all (transitive) predecessors l resp. $\perp$ that are in D.
- If all assignments are on decision level 0, then transitive predecessors of $\perp$ in D represent an IR.

Nonground Case:

- Optimized grounding algorithms inhibit reduction to the ground case.
- Remedy: Restrict the type of optimizations and give up completeness of IR computation (i.e., we do not always find an IR).

4. Trans-Unit (TU-)Propagation

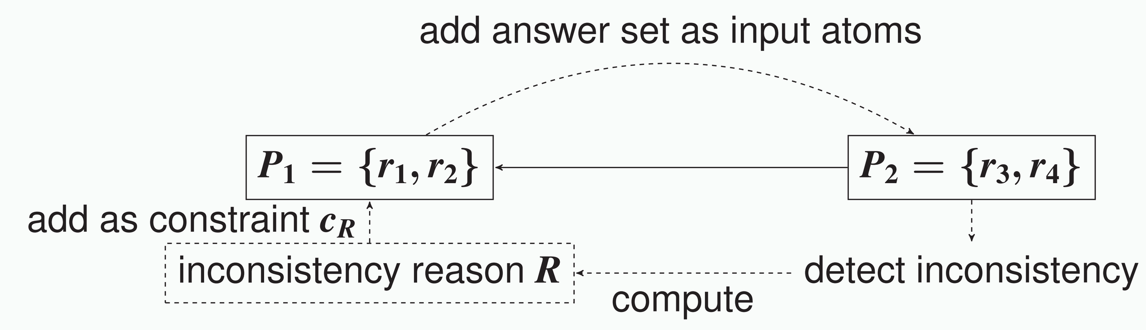
Existing Evaluation Approaches and Observations:

- Grounding P as a whole needs exponentially many external atom calls.
- Program splitting based on acyclic evaluation graph [Eiter et al., 2016] inhibits effective conflict-driven learning.
- $\Rightarrow$ There is either a grounding or a solving bottleneck.

Example

- Form a committee of employees; some pairs of persons are forbidden.
- Its competences depend nonmonotonically on the persons involved.
- Constraints define the competences the committee should have.

$$\begin{array}{rl} P = \{ r_1 : & in(X) \vee out(X) \leftarrow person(X). \\ r_2 : & \leftarrow in(X), in(Y), conflict(X, Y). \\ r_3 : & comp(X) \leftarrow \&competences[in](X). \\ r_4 : & \leftarrow \mathbf{not}\, comp(technical), \mathbf{not}\, comp(financial).\} \end{array}$$



TU-Propagation: Propagate an IR $R = (R^+, R^-)$ of a later program component as constraint $c_R = \leftarrow R^+, \{\mathbf{not}\, a \mid a \in R^-\}$ to predecessors.

5. Implementation and Experiments

We implemented tu-propagation in the **DLVHEX** solver and compared it to:

- monolithic evaluation as a single program (grounding bottleneck),
- splitting approach (solving bottleneck).

We considered several benchmark problems, including:

1. Configuration Problem:

size	monolithic				splitting				tu-propagation			
	total		ground	solve	total		ground	solve	total		ground	solve
8	0.20	(0)	0.05	0.04	0.49	(0)	0.15	0.20	0.19	(0)	<0.005	0.02
10	0.45	(0)	0.22	0.10	2.17	(0)	0.81	1.11	0.33	(0)	<0.005	0.05
12	1.51	(0)	0.90	0.44	9.50	(0)	3.59	5.17	0.86	(0)	0.01	0.10
14	4.84	(0)	3.68	0.82	44.23	(0)	16.58	24.68	1.48	(0)	0.02	0.20
16	19.52	(0)	16.55	2.08	217.46	(0)	86.17	119.93	3.80	(0)	0.07	0.57
18	143.09	(3)	68.89	10.89	300.00	(10)	122.23	165.44	44.76	(1)	0.43	6.29
20	300.00	(10)	300.00	n/a	300.00	(10)	126.40	162.73	37.44	(0)	0.36	3.44
22	300.00	(10)	300.00	n/a	300.00	(10)	122.41	165.68	224.83	(6)	1.03	11.74

2. Diagnosis Problem:

size	monolithic				splitting				tu-propagation			
	total		ground	solve	total		ground	solve	total		ground	solve
5	0.46	(0)	0.15	0.25	0.16	(0)	0.05	0.02	0.36	(0)	0.04	0.04
10	10.16	(0)	5.94	5.08	2.86	(0)	2.29	0.67	6.38	(0)	1.01	0.89
15	276.19	(5)	258.17	40.11	121.73	(1)	96.76	22.94	105.43	(3)	15.06	13.15
20	300.00	(10)	300.00	n/a	294.97	(9)	253.97	53.55	169.56	(5)	26.54	18.64
25	300.00	(10)	300.00	n/a	300.00	(10)	270.64	45.78	187.90	(5)	29.84	18.08
30	300.00	(10)	300.00	n/a	300.00	(10)	273.50	42.07	226.51	(6)	34.92	20.24
35	300.00	(10)	300.00	n/a	300.00	(10)	276.70	37.01	299.71	(9)	44.17	23.71

6. Conclusion and Outlook

Main results:

- Novel evaluation algorithm for HEX-programs and implementation.
- Experiments show a significant (up to exponential) speedup.

Future work:

- Generalization of tu-propagation from IRs to other learned nogoods.
- Exploit structure of the program to improve IR computation.

7. References

- Eiter, T., Fink, M., Ianni, G., Krennwallner, T., Redl, C., and Schüller, P. (2016).
A model building framework for answer set programming with external computations.
Theory and Practice of Logic Programming, 16(4):418–464.
- Eiter, T., Fink, M., Krennwallner, T., Redl, C., and Schüller, P. (2014).
Efficient HEX-program evaluation based on unfounded sets.
Journal of Artificial Intelligence Research, 49:269–321.
- Redl, C. (2017).
Explaining inconsistency in answer set programs and extensions.
In *Proceedings of the 14th International Conference on Logic Programming and Nonmonotonic Reasoning*.
To appear.

Der Wissenschaftsfonds.