Answer Set Solving in Practice

Torsten Schaub[1]
University of Potsdam
torsten@cs.uni-potsdam.de

Potassco

[1]Standing on the shoulders of a great research group and community!

Heuristic-driven solving: Overview

Potassco

Outline

Motivation

- Observation Sometimes it is advantageous to take a more application-oriented approach by including domain-specific information
  - domain-specific knowledge can be added for improving propagation
  - domain-specific heuristics can be used for making better choices
- Idea Incorporation of domain-specific heuristics by extending
  - input language and/or solver options for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics

Potassco

Motivation

- Observation Sometimes it is advantageous to take a more application-oriented approach by including domain-specific information
  - domain-specific knowledge can be added for improving propagation
  - domain-specific heuristics can be used for making better choices
- Idea Incorporation of domain-specific heuristics by extending
  - input language and/or solver options for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics

Potassco

Motivation

- Observation Sometimes it is advantageous to take a more application-oriented approach by including domain-specific information
  - domain-specific knowledge can be added for improving propagation
  - domain-specific heuristics can be used for making better choices
- Idea Incorporation of domain-specific heuristics by extending
  - input language and/or solver options for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics

CDCL-style solving

```
loop
    propagate                          // deterministically assign literals
    if no conflict then
        if all variables assigned then return solution
        else decide                    // non-deterministically assign some literal
    else
        if top-level conflict then return unsatisfiable
        else
            analyze                    // analyze conflict and add conflict constraint
            backjump                   // unassign literals until conflict constraint is unit
```

Inside *decide*

- Basic concepts
  - Atoms, $\mathcal{A}$
  - Assignments, $A : \mathcal{A} \to \{\boldsymbol{T}, \boldsymbol{F}\}$

    $A^{\boldsymbol{T}} = \{a \in \mathcal{A} \mid \boldsymbol{T}a \in A\}$ and $A^{\boldsymbol{F}} = \{a \in \mathcal{A} \mid \boldsymbol{F}a \in A\}$
- Heuristic functions

  $$h : \mathcal{A} \to [0, +\infty) \quad \text{and} \quad s : \mathcal{A} \to \{\boldsymbol{T}, \boldsymbol{F}\}$$
- Algorithmic scheme

  1. $h(a) := \alpha \times h(a) + \beta(a)$ for each $a \in \mathcal{A}$
  2. $U := \mathcal{A} \setminus (A^{\boldsymbol{T}} \cup A^{\boldsymbol{F}})$
  3. $C := \mathit{argmax}_{a \in U} h(a)$
  4. $a := \tau(C)$
  5. $A := A \cup \{a \mapsto s(a)\}$

Potassco

Inside *decide*

- Basic concepts
  - Atoms, $\mathcal{A}$
  - Assignments, $A : \mathcal{A} \to \{\boldsymbol{T}, \boldsymbol{F}\}$

    $A^{\boldsymbol{T}} = \{a \in \mathcal{A} \mid \boldsymbol{T}a \in A\}$ and $A^{\boldsymbol{F}} = \{a \in \mathcal{A} \mid \boldsymbol{F}a \in A\}$
- Heuristic functions

  $$h : \mathcal{A} \to [0, +\infty) \quad \text{and} \quad s : \mathcal{A} \to \{\boldsymbol{T}, \boldsymbol{F}\}$$
- Algorithmic scheme

  1. $h(a) := \alpha \times h(a) + \beta(a)$ for each $a \in \mathcal{A}$
  2. $U := \mathcal{A} \setminus (A^{\boldsymbol{T}} \cup A^{\boldsymbol{F}})$
  3. $C := \mathit{argmax}_{a \in U} h(a)$
  4. $a := \tau(C)$
  5. $A := A \cup \{a \mapsto s(a)\}$

Potassco

Inside *decide*

- Basic concepts
  - Atoms, $\mathcal{A}$
  - Assignments, $A : \mathcal{A} \to \{\boldsymbol{T}, \boldsymbol{F}\}$

    $A^{\boldsymbol{T}} = \{a \in \mathcal{A} \mid \boldsymbol{T}a \in A\}$ and $A^{\boldsymbol{F}} = \{a \in \mathcal{A} \mid \boldsymbol{F}a \in A\}$
- Heuristic functions

$$h : \mathcal{A} \to [0, +\infty) \quad \text{and} \quad s : \mathcal{A} \to \{\boldsymbol{T}, \boldsymbol{F}\}$$

- Algorithmic scheme

1. $h(a) := \alpha \times h(a) + \beta(a)$ for each $a \in \mathcal{A}$
2. $U := \mathcal{A} \setminus (A^{\boldsymbol{T}} \cup A^{\boldsymbol{F}})$
3. $C := \mathit{argmax}_{a \in U} h(a)$
4. $a := \tau(C)$
5. $A := A \cup \{a \mapsto s(a)\}$

Potassco

Inside *decide*

- Basic concepts
  - Atoms, $\mathcal{A}$
  - Assignments, $A : \mathcal{A} \to \{\boldsymbol{T}, \boldsymbol{F}\}$

    $A^{\boldsymbol{T}} = \{a \in \mathcal{A} \mid \boldsymbol{T}a \in A\}$ and $A^{\boldsymbol{F}} = \{a \in \mathcal{A} \mid \boldsymbol{F}a \in A\}$
- Heuristic functions

$$h : \mathcal{A} \to [0, +\infty) \quad \text{and} \quad s : \mathcal{A} \to \{\boldsymbol{T}, \boldsymbol{F}\}$$

- Algorithmic scheme

1. $h(a) := \alpha \times h(a) + \beta(a)$ for each $a \in \mathcal{A}$
2. $U := \mathcal{A} \setminus (A^{\boldsymbol{T}} \cup A^{\boldsymbol{F}})$
3. $C := \mathit{argmax}_{a \in U} h(a)$
4. $a := \tau(C)$
5. $A := A \cup \{a \mapsto s(a)\}$

Potassco

Outline

Outline

Heuristic language

- Heuristic directive

  `#heuristic` a `:` $l_1, \ldots, l_n$. $[k@p, m]$

  where
  - a is an atom, and $l_1, \ldots, l_n$ are literals
  - k and p are integers
  - m is a heuristic modifier

- Heuristic modifiers

`init`	for initializing the heuristic value of a with k
`factor`	for amplifying the heuristic value of a by factor k
`level`	for ranking all atoms; the rank of a is k
`sign`	for attributing the sign of k as truth value to a

- Example

```
#heuristic occurs(A,T) : action(A), time(T). [T, factor]
```

Potassco

Heuristic language

- Heuristic directive

  $\#\texttt{heuristic}\ a\ :\ l_1, \ldots, l_n.\ [k@p, m]$

  where
  - a is an atom, and $l_1, \ldots, l_n$ are literals
  - k and p are integers
  - m is a heuristic modifier
- Heuristic modifiers

`init`	for initializing the heuristic value of a with k
`factor`	for amplifying the heuristic value of a by factor k
`level`	for ranking all atoms; the rank of a is k
`sign`	for attributing the sign of k as truth value to a

- Example

```
#heuristic occurs(A,T) : action(A), time(T). [T, factor]
```

Potassco

Heuristic language

- Heuristic directive

  `#heuristic` a `:` $l_1, \ldots, l_n$. $[k@p, m]$

  where
  - a is an atom, and $l_1, \ldots, l_n$ are literals
  - k and p are integers
  - m is a heuristic modifier

- Heuristic modifiers

`init`	for initializing the heuristic value of a with k
`factor`	for amplifying the heuristic value of a by factor k
`level`	for ranking all atoms; the rank of a is k
`sign`	for attributing the sign of k as truth value to a
`true/false`	combine `level` and `sign`

- Example

```
#heuristic occurs(A,T) : action(A), time(T). [T, factor]
```

Heuristic language

- Heuristic directive

  $\#\texttt{heuristic}\ a\ \texttt{:}\ l_1, \ldots, l_n.\ [k@p, m]$

  where
  - a is an atom, and $l_1, \ldots, l_n$ are literals
  - k and p are integers
  - m is a heuristic modifier
- Heuristic modifiers

`init`	for initializing the heuristic value of a with k
`factor`	for amplifying the heuristic value of a by factor k
`level`	for ranking all atoms; the rank of a is k
`sign`	for attributing the sign of k as truth value to a

- Example

```
#heuristic occurs(A,T) : action(A), time(T). [T, factor]
```

Potassco

Heuristic language

- Heuristic directive

  $\texttt{\#heuristic}\ a\ \texttt{:}\ l_1, \ldots, l_n.\ [k@p, m]$

  where
  - a is an atom, and $l_1, \ldots, l_n$ are literals
  - k and p are integers
  - m is a heuristic modifier
- Heuristic modifiers

`init`	for initializing the heuristic value of a with k
`factor`	for amplifying the heuristic value of a by factor k
`level`	for ranking all atoms; the rank of a is k
`sign`	for attributing the sign of k as truth value to a

- Example

```
#heuristic occurs(mv,5) : action(mv), time(5). [5, factor]
```

Potassco

Simple STRIPS planning

```
time(1..k).

holds(P,0) :- init(P).

{ occ(A,T) : action(A) } = 1 :- time(T).
:- occ(A,T), pre(A,F), not holds(F,T-1).

holds(F,T) :- occ(A,T), add(A,F).
holds(F,T) :- holds(F,T-1), time(T), not occ(A,T) : del(A,F).

:- query(F), not holds(F,k).
```

Simple STRIPS planning

```
time(1..k).

holds(P,0) :- init(P).

{ occ(A,T) : action(A) } = 1 :- time(T).
:- occ(A,T), pre(A,F), not holds(F,T-1).

holds(F,T) :- occ(A,T), add(A,F).
holds(F,T) :- holds(F,T-1), time(T), not occ(A,T) : del(A,F).

:- query(F), not holds(F,k).

#heuristic occurs(A,T) : action(A), time(T). [2, factor]
```

Potassco

Simple STRIPS planning

```
time(1..k).

holds(P,0) :- init(P).

{ occ(A,T) : action(A) } = 1 :- time(T).
:- occ(A,T), pre(A,F), not holds(F,T-1).

holds(F,T) :- occ(A,T), add(A,F).
holds(F,T) :- holds(F,T-1), time(T), not occ(A,T) : del(A,F).

:- query(F), not holds(F,k).

#heuristic occurs(A,T) : action(A), time(T). [1, level]
```

Potassco

Simple STRIPS planning

```
time(1..k).

holds(P,0) :- init(P).

{ occ(A,T) : action(A) } = 1 :- time(T).
:- occ(A,T), pre(A,F), not holds(F,T-1).

holds(F,T) :- occ(A,T), add(A,F).
holds(F,T) :- holds(F,T-1), time(T), not occ(A,T) : del(A,F).

:- query(F), not holds(F,k).

#heuristic occurs(A,T) : action(A), time(T). [T, factor]
```

Potassco

Simple STRIPS planning

```
time(1..k).

holds(P,0) :- init(P).

{ occ(A,T) : action(A) } = 1 :- time(T).
:- occ(A,T), pre(A,F), not holds(F,T-1).

holds(F,T) :- occ(A,T), add(A,F).
holds(F,T) :- holds(F,T-1), time(T), not occ(A,T) : del(A,F).

:- query(F), not holds(F,k).

#heuristic holds(F,T-1) :     holds(F,T). [t-T+1, true]
#heuristic holds(F,T-1) : not holds(F,T)  [t-T+1, false]
                          fluent(F), time(T).
```

Potassco

Outline

Potassco

Heuristic options

- Alternative for specifying structure-oriented heuristics in *clasp*

```
--dom-mod=<arg> : Default modification for
                  domain heuristic
  <arg>: <mod>[,<pick>]
    <mod>  : Modifier
             {1=level|2=pos|3=true|4=neg|
              5=false|6=init|7=factor}
    <pick> : Apply <mod> to
             {0=all|1=scc|2=hcc|4=disj|
              8=min|16=show} atoms
```

Engage heuristic modifications (in both settings!)

```
--heuristic=Domain
```

Heuristic options

- Alternative for specifying structure-oriented heuristics in *clasp*

```
--dom-mod=<arg> : Default modification for
                  domain heuristic
   <arg>: <mod>[,<pick>]
     <mod>  : Modifier
              {1=level|2=pos|3=true|4=neg|
               5=false|6=init|7=factor}
     <pick> : Apply <mod> to
              {0=all|1=scc|2=hcc|4=disj|
               8=min|16=show} atoms
```

- Engage heuristic modifications (in both settings!)

```
--heuristic=Domain
```

Heuristic options

- Alternative for specifying structure-oriented heuristics in *clasp*

```
--dom-mod=<arg> : Default modification for
                  domain heuristic
   <arg>: <mod>[,<pick>]
     <mod>  : Modifier
              {1=level|2=pos|3=true|4=neg|
               5=false|6=init|7=factor}
     <pick> : Apply <mod> to
              {0=all|1=scc|2=hcc|4=disj|
               8=min|16=show} atoms
```

- Engage heuristic modifications (in both settings!)

```
--heuristic=Domain
```

Potassco

Inclusion-minimal stable models

- Consider a logic program containing a mimimize statement of form
  - `#minimize{`$a_1, \ldots, a_n$`}`
- Computing one inclusion-minimal stable model can be done either via
  - `#heuristic` a_i `[1,false].` for $i = 1, \ldots, n$, or
  - `--dom-mod=5,16`
- Computing all inclusion-minimal stable model can be done
  - by adding `--enum-mod=domRec` to the two options

Potassco

Inclusion-minimal stable models

- Consider a logic program containing a mimimize statement of form
  - `#minimize`$\{a_1, \ldots, a_n\}$
- Computing one inclusion-minimal stable model can be done either via
  - `#heuristic` a_i `[1,false].` for $i = 1, \ldots, n$, or
  - `--dom-mod=5,16`
- Computing all inclusion-minimal stable model can be done
  - by adding `--enum-mod=domRec` to the two options

Inclusion-minimal stable models

- Consider a logic program containing a mimimize statement of form
  - `#minimize`$\{a_1, \ldots, a_n\}$
- Computing one inclusion-minimal stable model can be done either via
  - `#heuristic` a_i `[1,false].` for $i = 1, \ldots, n$, or
  - `--dom-mod=5,16`
- Computing all inclusion-minimal stable model can be done
  - by adding `--enum-mod=domRec` to the two options

Outline

Summary

- TBF

Auf Wiedersehen!

"Tomorrow isn't staying out
I'll be back, without a doubt!"

Pink panther

Potassco

Bibliography

- The following list of references is compiled from the open source bibliography available at

  `https://github.com/krr-up/bibliography`

- Feel free to submit corrections via pull requests!

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] M. Alviano et al. "The ASP System DLV2". In: *Proceedings of the Fourteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'17)*. Ed. by M. Balduccini and T. Janhunen. Vol. 10377. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2017, pp. 215–221.

[3] M. Alviano et al. "The Fourth Answer Set Programming Competition: Preliminary Report". In: *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13)*. Ed. by P. Cabalar and T. Son. Vol. 8148. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2013, pp. 42–53.

[4] M. Alviano et al. "WASP: A Native ASP Solver Based on Constraint Learning". In: *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13)*.

Potassco

Ed. by P. Cabalar and T. Son. Vol. 8148. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2013, pp. 54–66.

[5] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[6] C. Baral, G. Brewka, and J. Schlipf, eds. *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*. Vol. 4483. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2007.

[7] C. Baral and M. Gelfond. "Logic Programming and Knowledge Representation". In: *Journal of Logic Programming* 12 (1994), pp. 1–80.

[8] P. Borchert et al. "Towards Systematic Benchmarking in Answer Set Programming: The Dagstuhl Initiative". In: *Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'04)*. Ed. by V. Lifschitz and I. Niemelä. Vol. 2923. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2004, pp. 3–7.

Potassco

[9] G. Brewka, T. Eiter, and M. Truszczynski. "Answer Set Programming: An Introduction to the Special Issue". In: *AI Magazine* 37.3 (2016), pp. 5–6.

[10] G. Brewka, T. Eiter, and M. Truszczyński. "Answer set programming at a glance". In: *Communications of the ACM* 54.12 (2011), pp. 92–103.

[11] P. Cabalar and T. Son, eds. *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13)*. Vol. 8148. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2013.

[12] F. Calimeri et al. "ASP-Core-2 Input Language Format". In: *Theory and Practice of Logic Programming* 20.2 (2019), pp. 294–309.

[13] F. Calimeri et al. "ASP-Core-2 Input Language Format". In: *Theory and Practice of Logic Programming* 20.2 (2020), pp. 294–309.

[14] F. Calimeri et al. "I-DLV: The new intelligent grounder of DLV". In: *Intelligenza Artificiale* 11.1 (2017), pp. 5–20.

[15] F. Calimeri et al. "The Design of the Fifth Answer Set Programming Competition". In: *Technical Communications of the Thirtieth International Conference on Logic Programming (ICLP'14)*. Ed. by M. Leuschel and T. Schrijvers. Vol. arXiv:1405.3710v4. Theory and Practice of Logic Programming, Online Supplement. 2014. URL: http://arxiv.org/abs/1405.3710v4.

[16] F. Calimeri et al. "The Third Answer Set Programming Competition: Preliminary Report of the System Competition Track". In: *Proceedings of the Eleventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*. Ed. by J. Delgrande and W. Faber. Vol. 6645. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2011, pp. 388–403.

[17] M. D'Agostino et al., eds. *Handbook of Tableau Methods*. Kluwer Academic Publishers, 1999.

[18] J. Delgrande and W. Faber, eds. *Proceedings of the Eleventh International Conference on Logic Programming and Nonmonotonic*

Reasoning (LPNMR'11). Vol. 6645. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2011.

[19] M. Denecker et al. "The Second Answer Set Programming Competition". In: *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*. Ed. by E. Erdem, F. Lin, and T. Schaub. Vol. 5753. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2009, pp. 637–654.

[20] T. Eiter, G. Ianni, and T. Krennwallner. "Answer Set Programming: A Primer". In: *Fifth International Reasoning Web Summer School (RW'09)*. Ed. by S. Tessaris et al. Vol. 5689. Lecture Notes in Computer Science. Slides at http://www.kr.tuwien.ac.at/staff/tkren/pub/2009/rw2009-lecture.zip. Springer-Verlag, 2009, pp. 40–110. URL: http://www.kr.tuwien.ac.at/staff/tkren/pub/2009/rw2009-asp.pdf.

[21] E. Erdem, F. Lin, and T. Schaub, eds. *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic*

Reasoning (LPNMR'09). Vol. 5753. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2009.

[22] M. Fitting. "A Kripke-Kleene Semantics for Logic Programs". In: *Journal of Logic Programming* 2.4 (1985), pp. 295–312.

[23] M. Fitting. "Fixpoint semantics for logic programming: A survey". In: *Theoretical Computer Science* 278.1-2 (2002), pp. 25–51.

[24] M. Gebser, B. Kaufmann, and T. Schaub. "Conflict-Driven Answer Set Solving: From Theory to Practice". In: *Artificial Intelligence* 187-188 (2012), pp. 52–89.

[25] M. Gebser, T. Schaub, and S. Thiele. "Gringo: A New Grounder for Answer Set Programming". In: *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*. Ed. by C. Baral, G. Brewka, and J. Schlipf. Vol. 4483. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2007, pp. 266–271.

[26] M. Gebser et al. "Abstract Gringo". In: *Theory and Practice of Logic Programming* 15.4-5 (2015), pp. 449–463.

Potassco

[27] M. Gebser et al. "Advances in gringo Series 3". In: *Proceedings of the Eleventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*. Ed. by J. Delgrande and W. Faber. Vol. 6645. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2011, pp. 345–351.

[28] M. Gebser et al. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

[29] M. Gebser et al. "Conflict-Driven Answer Set Solving". In: *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*. Ed. by M. Veloso. AAAI/MIT Press, 2007, pp. 386–392.

[30] M. Gebser et al. "Multi-shot ASP solving with clingo". In: *Theory and Practice of Logic Programming* 19.1 (2019), pp. 27–82.

[31] M. Gebser et al. "On the Input Language of ASP Grounder Gringo". In: *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*. Ed. by

E. Erdem, F. Lin, and T. Schaub. Vol. 5753. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2009, pp. 502–508.

[32] M. Gebser et al. *Potassco User Guide*. 2nd ed. University of Potsdam. 2015. URL: http://potassco.org.

[33] M. Gebser et al. "The First Answer Set Programming System Competition". In: *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*. Ed. by C. Baral, G. Brewka, and J. Schlipf. Vol. 4483. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2007, pp. 3–17.

[34] M. Gelfond. "Answer Sets". In: *Handbook of Knowledge Representation*. Ed. by V. Lifschitz, F. van Harmelen, and B. Porter. Elsevier Science, 2008. Chap. 7, pp. 285–316.

[35] M. Gelfond and Y. Kahl. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press, 2014.

[36] M. Gelfond and N. Leone. "Logic programming and knowledge representation — the A-Prolog perspective". In: *Artificial Intelligence* 138.1-2 (2002), pp. 3–38.

[37] M. Gelfond and V. Lifschitz. "Logic Programs with Classical Negation". In: *Proceedings of the Seventh International Conference on Logic Programming (ICLP'90)*. Ed. by D. Warren and P. Szeredi. MIT Press, 1990, pp. 579–597.

[38] M. Gelfond and V. Lifschitz. "The Stable Model Semantics for Logic Programming". In: *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*. Ed. by R. Kowalski and K. Bowen. MIT Press, 1988, pp. 1070–1080.

[39] R. Kaminski, T. Schaub, and P. Wanko. "A Tutorial on Hybrid Answer Set Solving with clingo". In: *Proceedings of the Thirteenth International Summer School of the Reasoning Web*. Ed. by G. Ianni et al. Vol. 10370. Lecture Notes in Computer Science. Springer-Verlag, 2017, pp. 167–203.

[40] J. Lee. "A Model-Theoretic Counterpart of Loop Formulas". In: *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*. Ed. by L. Kaelbling and A. Saffiotti. Professional Book Center, 2005, pp. 503–508.

[41] N. Leone et al. "The DLV System for Knowledge Representation and Reasoning". In: *ACM Transactions on Computational Logic* 7.3 (2006), pp. 499–562.

[42] V. Lifschitz. *Answer Set Programming*. Springer-Verlag, 2019.

[43] V. Lifschitz. "Answer set programming and plan generation". In: *Artificial Intelligence* 138.1-2 (2002), pp. 39–54.

[44] V. Lifschitz. "Introduction to answer set programming". Unpublished draft. 2004. URL: http://www.cs.utexas.edu/users/vl/papers/esslli.ps.

[45] V. Lifschitz. "Thirteen Definitions of a Stable Model". In: *Fields of Logic and Computation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*. Ed. by A. Blass, N. Dershowitz, and

W. Reisig. Vol. 6300. Lecture Notes in Computer Science. Springer-Verlag, 2010, pp. 488–503.

[46] V. Lifschitz. "Twelve Definitions of a Stable Model". In: *Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08)*. Ed. by M. Garcia de la Banda and E. Pontelli. Vol. 5366. Lecture Notes in Computer Science. Springer-Verlag, 2008, pp. 37–51.

[47] V. Marek and M. Truszczyński. *Nonmonotonic logic: context-dependent reasoning*. Artifical Intelligence. Springer-Verlag, 1993.

[48] V. Marek and M. Truszczyński. "Stable models and an alternative logic programming paradigm". In: *The Logic Programming Paradigm: a 25-Year Perspective*. Ed. by K. Apt et al. Springer-Verlag, 1999, pp. 375–398.

[49] I. Niemelä. "Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm". In: *Annals of Mathematics and Artificial Intelligence* 25.3-4 (1999), pp. 241–273.

[50] I. Niemelä and P. Simons. "Efficient Implementation of the Well-founded and Stable Model Semantics". In: *Proceedings of the Joint International Conference and Symposium on Logic Programming*. Ed. by M. Maher. MIT Press, 1996, pp. 289–303.

[51] D. Saccá and C. Zaniolo. "Stable Models and Non-Determinism in Logic Programs with Negation". In: *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. ACM Press, 1990, pp. 205–217.

[52] T. Schaub and S. Woltran. "Answer set programming unleashed!" In: *Künstliche Intelligenz* 32.2-3 (2018), pp. 105–108.

[53] T. Schaub and S. Woltran. "Special Issue on Answer Set Programming". In: *Künstliche Intelligenz* 32.2-3 (2018), pp. 101–103.

[54] P. Simons, I. Niemelä, and T. Soininen. "Extending and implementing the stable model semantics". In: *Artificial Intelligence* 138.1-2 (2002), pp. 181–234.

[55] T. Syrjänen. "Omega-Restricted Logic Programs". In: *Proceedings of the Sixth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'01)*. Ed. by T. Eiter, W. Faber, and M. Truszczyński. Vol. 2173. Lecture Notes in Computer Science. Springer-Verlag, 2001, pp. 267–279.

[56] M. Truszczyński. "An introduction to the stable and well-founded semantics of logic programs". In: *Declarative Logic Programming: Theory, Systems, and Applications*. Ed. by M. Kifer and Y. Liu. ACM / Morgan & Claypool, 2018, pp. 121–177.

[57] A. Van Gelder, K. Ross, and J. Schlipf. "The Well-Founded Semantics for General Logic Programs". In: *Journal of the ACM* 38.3 (1991), pp. 620–650.