

Rushing and Strolling among Answer Sets – Navigation Made Easy*

Johannes Klaus Fichte¹, Sarah Alice Gaggl², and Dominik Rusovac²

¹TU Wien, Austria, johannes.fichte@tuwien.ac.at

²TU Dresden, Germany, firstname.lastname@tu-dresden.de

Abstract

Answer set programming (ASP) is a popular declarative programming paradigm with a wide range of applications in artificial intelligence. Oftentimes, when modeling an AI problem with ASP, and in particular when we are interested beyond simple search for optimal solutions, an actual solution, differences between solutions, or number of solutions of the ASP program matter. For example, when a user aims to identify a specific answer set according to her needs, or requires the total number of diverging solutions to comprehend probabilistic applications such as reasoning in medical domains. Then, there are only certain problem specific and handcrafted encoding techniques available to navigate the solution space of ASP programs, which is oftentimes not enough. In this paper, we propose a formal and general framework for interactive navigation towards desired subsets of answer sets analogous to faceted browsing. Our approach enables the user to explore the solution space by consciously zooming in or out of sub-spaces of solutions at a certain configurable pace. We illustrate that weighted faceted navigation is computationally hard. Finally, we provide an implementation of our approach that demonstrates the feasibility of our framework for incomprehensible solution spaces.

Introduction

Answer set programming (ASP) is a declarative programming paradigm, which has its roots in logic programming and nonmonotonic reasoning. It is widely used for knowledge representation and problem solving Brewka *et al.* (2011); Eiter *et al.* (2009); Gebser *et al.* (2012). In ASP, a problem is encoded as a set of rules (logic program) and is evaluated under stable model semantics Gelfond and Lifschitz (1988, 1991), using solvers such as `clingo` Gebser *et al.* (2011a, 2014), `WASP` Alviano *et al.* (2015), or `DLV` Alviano *et al.* (2017). Then, answer sets represent solutions to the modeled problem.

Oftentimes when modeling with ASP, the number of solutions of the resulting program can be quite high. This is not necessarily a problem when searching for a few solutions, e.g., optimal solutions Gebser *et al.* (2011b); Alviano and Dodaro (2016a) or when incorporating preferences Brewka (2004); Brewka *et al.* (2015a,b); Alviano *et al.* (2018). However, there are many situations where reasoning goes beyond simple search for one answer set, for example, planning when certain routes are gradually forbidden Son *et al.* (2016), finding diverging solutions Everardo (2017); Everardo *et al.* (2019), reasoning in probabilistic applications Lee *et al.* (2017), or debugging answer sets Oetsch *et al.* (2018); Dodaro *et al.* (2019); Vos *et al.* (2012); Shchekotykhin (2015); Gebser *et al.* (2008).

Now, if the user is interested in more than a few solutions to gradually identify specific answer sets, tremendous solution spaces can easily become infeasible to comprehend. In fact, it might not even be

*This is the authors' self-archived copy, including proofs, of a paper that has been accepted for publication at AAAI-22.

possible to compute all solutions in reasonable time. Examples where we easily see large solution spaces are configuration problems Soininen and Niemelä (1999); Soininen *et al.* (2001); Tiihonen *et al.* (2003), such as for instance PC configuration, and planning problems Dimopoulos *et al.* (1997); Lifschitz (1999); Nogueira *et al.* (2001). Let us consider a simple example to illustrate the use of navigation in ASP.

Example 1. Consider an online shopping situation where we have a knowledge base on clothes and some rules which specify which combinations would suit well or not.

$$\begin{aligned} & \{ \{ \text{outfit}(X, Y) : \text{clothes}(X, Y) \}; \\ & \quad \leftarrow \text{outfit}(X, Y1), \\ & \quad \quad \text{outfit}(X, Y2), Y1 \neq Y2; \\ & \text{occasion}(\text{vancouver}) \leftarrow \text{outfit}(\text{jacket}, \dots); \\ & \text{occasion}(\text{conference}) \leftarrow \text{outfit}(\text{suit}, Y), Y \neq \text{yellow}; \\ & \text{occasion}(\text{wistler}) \leftarrow \text{outfit}(\text{boots}, \dots) \dots \} \end{aligned}$$

Together with input facts from a clothes database like $\text{clothes}(\text{jacket}, \text{blue})$; $\text{clothes}(\text{shirt}, \text{red})$; \dots one easily obtains more than a million answer sets. Since Canada opened immigration for vaccinated persons, we actually might be able to travel to Vancouver. Say we zoom in on outfits including shorts, which leads to a rather small, but still incomprehensible sub-space of solutions. Imagine that most of the remaining outfits include chucks and a jacket. Say we want to inspect the most different outfits still remaining, then we aim to choose potential parts of our outfit that provide us with most diverse solutions. Now, we are almost good to go, seeking to find some final additions to our outfit quickly.

Our example illustrates that different solutions in ASP programs can easily be hard to comprehend. Problem specific, handcrafted encoding techniques to navigate the solution space can be quite tedious.

Instead, we propose a formal and general framework for interactive navigation towards desired subsets of answer sets analogous to faceted browsing in the field of information retrieval Tunkelang (2009). Our approach enables solution space exploration by consciously zooming in or out of sub-spaces of solutions at a certain configurable pace. To this end we introduce absolute and relative weights to quantify the size of the search space when reasoning under assumptions (facets). We formalize several kinds of search space navigation as goal-oriented and explore modes, and systematically compare the introduced weights regarding their usability for operations under natural properties splitting, reliability, preserving maximal sub-spaces (min-inline), and preserving minimal sub-spaces (max-inline). In addition, we illustrate the computational complexity for computing the weights. Finally, we provide an implementation on top of the solver `clingo` demonstrating the feasibility of our framework for incomprehensible solution spaces.

Related Work. Alrabbaa *et al.* (2018) proposed a framework in which solutions are systematically pruned with respect to facets (partial solutions). While this allows one to move within the answer set space, the user has absolutely no information on how big the effect of activating a facet is in advance, similar to assumptions in propositional satisfiability Eén and Sörensson (2003). We go far beyond and characterize the *weight* of a facet. This is useful to comprehend the effect of navigation steps on the size of the solution space. Additionally, this allows for zooming into or out of the solution space at a configurable pace. Debugging in answer sets has widely been investigated Oetsch *et al.* (2018); Dodaro *et al.* (2019); Vos *et al.* (2012); Shchekotykhin (2015); Gebser *et al.* (2008). However, we do not aim to correct ASP encodings. All answer sets which are reachable within the navigation are “original” answer sets, thus the adaptations we make during the navigation to the program, do not change the set of answer sets of the initial program. Justifications, which describe the support for the truth value of each atom, have been studied as a tool for reasoning and debugging El-Khatib *et al.* (2005). Probabilistic reasoning frameworks for logic programs were developed such as LP^{MLN} Lee *et al.* (2017), which define notions of probabilities in terms of

relative occurrences of stable models and their weights. Computing these probabilities (unless restricted to decision versions in terms of being different from zero) relates to counting probabilities under assumptions. Considering relative occurrences of stable models of weight one relates to search space exploration. However, probabilistic frameworks primarily address modeling conflicting information and reason about them. We assume large solution spaces and aim for navigating dynamically in the solution space.

Background

First, we recall basic notions of ASP, for further details on ASP we refer to standard texts Calimeri *et al.* (2020); Gebser *et al.* (2012). Then, we introduce fundamental notions of faceted navigation and computational complexity, respectively.

Answer Set Programming. By $\mathcal{A}(\Pi)$ we denote the set of (non-ground) *atoms* of a program Π . A literal is an atom $\alpha \in \mathcal{A}(\Pi)$ or its *default negation*, which refers to the absence of information, denoted by $\sim\alpha$. An atom α is a predicate $p(t_0, \dots, t_n)$ of arity $n \geq 0$ where each t_i for $0 \leq i \leq n$ is a *term*, i.e., either a variable or a constant. We say an atom $\alpha \in \mathcal{A}(\Pi)$ is *ground* if and only if α is variable-free. By $\text{Grd}(\mathcal{A}(\Pi))$ we denote ground atoms. A (disjunctive) logic program Π is a finite set of rules r of the form

$$\alpha_0 \mid \dots \mid \alpha_k \leftarrow \alpha_{k+1}, \dots, \alpha_m, \sim\alpha_{m+1}, \dots, \sim\alpha_n$$

where $0 \leq k \leq m \leq n$ and each $\alpha_i \in \mathcal{A}(\Pi)$ for $0 \leq i \leq n$. For a rule r we denote the head by $H(r) := \{\alpha_0, \dots, \alpha_k\}$, the body $B(r)$ consists of the positive body $B^+(r) := \{\alpha_{k+1}, \dots, \alpha_m\}$, and the negative body $B^-(r) := \{\alpha_{m+1}, \dots, \alpha_n\}$. If $B(r) = \emptyset$, we omit \leftarrow . A rule r where $H(r) = \emptyset$ is called *integrity constraint* and avoids that $B(r)$ is evaluated positively. By $\text{grd}(r)$ we denote the set of ground instances of some rule r , obtained by replacing all variables in r by ground terms. Accordingly, $\text{Grd}(\Pi) := \bigcup_{r \in \Pi} \text{grd}(r)$ denotes the ground instantiation of Π . Without any explicit contrary indication, throughout this paper, we use the term (logic) program to refer to grounded disjunctive programs where $\mathcal{A}(\Pi) = \text{Grd}(\mathcal{A}(\Pi))$. An interpretation $X \subseteq \mathcal{A}(\Pi)$ satisfies a rule $r \in \Pi$ if and only if $H(r) \cap X \neq \emptyset$ whenever $B^+(r) \subseteq X$ and $B^-(r) \cap X = \emptyset$. X satisfies Π , if X satisfies each rule $r \in \Pi$. An interpretation X is a *stable model* (also called *answer set*) of Π if and only if X is a subset-minimal model satisfying the Gelfond-Lifschitz reduct of Π with respect to X , defined as $\Pi_X := \{H(r) \leftarrow B^+(r) \mid X \cap B^-(r) = \emptyset, r \in \Pi\}$. By $\mathcal{AS}(\Pi)$ we denote the answer sets of Π . For computing facets, we rely on two notions of consequences of a program, namely, *brave* consequences $\mathcal{BC}(\Pi) := \bigcup \mathcal{AS}(\Pi)$ and *cautious* consequences $\mathcal{CC}(\Pi) := \bigcap \mathcal{AS}(\Pi)$.

Faceted Navigation. *Faceted answer set navigation* is characterized as a sequence of navigation steps restricting the solution space with respect to partial solutions. Those partial solutions, called *facets*, correspond to ground atoms of a program Π that are not contained in each solution. We denote the *facets* of Π by $\mathcal{F}(\Pi) := \mathcal{F}^+(\Pi) \cup \mathcal{F}^-(\Pi)$ where $\mathcal{F}^+(\Pi) := \mathcal{BC}(\Pi) \setminus \mathcal{CC}(\Pi)$ denotes *inclusive facets* and $\mathcal{F}^-(\Pi) := \{\bar{\alpha} \mid \alpha \in \mathcal{F}^+(\Pi)\}$ denotes *exclusive facets* of Π . We say an interpretation $X \subseteq \mathcal{A}(\Pi)$ satisfies an inclusive facet $f \in \mathcal{F}^+(\Pi)$, if $f \in X$, which we denote by $X \models f$, and it satisfies an exclusive facet $f \in \mathcal{F}^-(\Pi)$, if $f \notin X$.

A navigation step is a transition from one program to another, obtained by adding some integrity constraint that enforces the atom referred to by some inclusive or exclusive facet to be present or absent, respectively, throughout answer sets. By $ic(f)$ we denote the function that translates a facet $f \in \{\alpha, \bar{\alpha}\} \subseteq \mathcal{F}(\Pi)$ into a singleton program that contains its corresponding integrity constraint:

$$ic(f) := \begin{cases} \{\leftarrow \sim\alpha\}, & \text{if } f = \alpha; \\ \{\leftarrow \alpha\}, & \text{otherwise.} \end{cases}$$

Accordingly, a navigation step from Π to Π' is obtained by modifying Π such that $\Pi' = \Pi \cup ic(f)$. Faceted navigation w.r.t. some program Π is possible as long as $\mathcal{F}(\Pi) \neq \emptyset$. Alrabbaa *et al.* (2018) established that if $f \in \mathcal{F}(\Pi)$, then $\Pi' := \Pi \cup ic(f)$ is satisfiable and $\mathcal{AS}(\Pi') = \{X \in \mathcal{AS}(\Pi) \mid X \models f\}$. When referring to $\mathcal{AS}(\Pi)$ as a solution space, we refer to the topological space induced by $2^{\mathcal{AS}(\Pi)}$ on $\mathcal{AS}(\Pi)$. Thus, answer set navigation means choosing among subsets of answer sets.

Computational Complexity. We assume that the reader is familiar with the main concepts of computational complexity theory Papadimitriou (1994); Arora and Barak (2009) and follows standard terminology in the area of counting complexity Durand *et al.* (2005); Hemaspaandra and Vollmer (1995). Recall that P and NP are the complexity classes of all deterministically and non-deterministically polynomial-time solvable decision problems Cook (1971), respectively. For a complexity class C, co-C denotes the class of all decision problems whose complement is in C. We are also interested in the polynomial hierarchy Stockmeyer and Meyer (1973); Stockmeyer (1976); Wrathall (1976) defined as follows: $\Delta_0^p := \Pi_0^p := \Sigma_0^p := P$ and $\Delta_i^p := P^{\Sigma_i^p}$, $\Sigma_i^p := NP^{\Sigma_i^p}$, $\Pi_i^p := coNP^{\Sigma_i^p}$ for $i > 0$ where C^D is the class C of decision problems augmented by an oracle for some complete problem in class D. Further, $PH := \bigcup_{k \in \mathbb{N}} \Delta_k^p$. Note that $NP = \Sigma_1^p$, $coNP = \Pi_1^p$, $\Sigma_2^p = NP^{NP}$, and $\Pi_2^p = coNP^{NP}$. If \mathcal{C} is a decision complexity class then $\# \cdot \mathcal{C}$ is the class of all counting problems whose witness function w satisfies (i) \exists polynomial p such that for all $y \in w(x)$, we have that $|y| \leq p(|x|)$, and (ii) the decision problem “given x and y , is $y \in w(x)$?” is in \mathcal{C} . A *witness* function is a function $w: \Sigma^* \rightarrow \mathcal{P}^{<\omega}(\Gamma^*)$, where Σ and Γ are alphabets, mapping to a finite subset of Γ^* . Such functions associate with the counting problem “given $x \in \Sigma^*$, find $|w(x)|$ ”.

Routes and Navigation Modes

We introduce *routes* as a notion for characterizing sequences of navigation steps.

Definition 1. A route δ is a finite sequence $\langle f_1, \dots, f_n \rangle$ of facets $f_i \in \mathcal{F}(\Pi)$ such that $0 \leq i \leq n \in \mathbb{N}$, denoting n arbitrary navigation steps over Π . We say δ is a subroute of δ' , denoted by $\delta \sqsubseteq \delta'$, whenever if $f_i \in \delta$, then $f_i \in \delta'$. We define $\Pi^\delta := \Pi \cup ic(f_1) \cup \dots \cup ic(f_n)$. By Δ^Π we denote all possible routes over $\mathcal{AS}(\Pi)$, including the empty route ϵ .

It is easy to see that any permutation of navigation steps of a fixed set of facets always leads to the same solutions. In general, different routes may lead to the same subset of answer sets. We say two routes $\delta, \delta' \in \Delta^\Pi$ are equivalent if and only if $\mathcal{AS}(\Pi^\delta) = \mathcal{AS}(\Pi^{\delta'})$. To ensure satisfiable programs, we aim to select so called *safe* routes. By $\Delta_s^\Pi := \{\delta \in \Delta^\Pi \mid \mathcal{AS}(\Pi^\delta) \neq \emptyset\}$ we define *safe routes* over $\mathcal{AS}(\Pi)$. Once an unsafe route is taken, some sort of *redirection*, which relates to the notion of *correction sets* Alrabbaa *et al.* (2018), i.e., a route obtained by retracting conflicting facets, is required to continue navigation. For a program Π , $\delta \in \Delta^\Pi$ and $f \in \mathcal{F}(\Pi)$. We denote all *redirections* of δ with respect to f by $\mathcal{R}(\delta, f) := \{\delta' \sqsubseteq \delta \mid f \in \delta', \mathcal{AS}(\Pi^{\delta'}) \neq \emptyset\} \cup \{\epsilon\}$. The following example illustrates faceted navigation.

Example 2. Consider program $\Pi_1 = \{a \mid b; c \mid d \leftarrow b; e\}$. It is easy to observe that the answer sets are $\mathcal{AS}(\Pi_1) = \{\{a, e\}, \{b, c, e\}, \{b, d, e\}\}$. Thus, we can choose from facets $\mathcal{F}(\Pi_1) = \{a, b, c, d, \bar{a}, \bar{b}, \bar{c}, \bar{d}\}$. As illustrated in Figure 1, if we activate facet a we land at $\mathcal{AS}(\Pi_1^{(a)}) = \{\{a, e\}\}$. Activating b on $\langle a \rangle$ gives $\mathcal{AS}(\Pi_1^{(a,b)}) = \emptyset$. To redirect $\langle a, b \rangle$ we can choose from $\mathcal{R}(\langle a, b \rangle, b) = \{\langle b \rangle\}$.

We consider two more notions for identifying routes that point to a unique solution. A set of facets is a *delimitation*, if any safe route constructible thereof leads to a unique answer set. This means that any further step would lead to an unsafe route.

Definition 2. Let Π be a program and $F, F' \subseteq \mathcal{F}(\Pi)$ such that $F := \{f_1, \dots, f_n\}$. We define $\tau(F)$ as all permutations of $\delta := \langle f_1, \dots, f_n \rangle$ and say F is *delimiting* with respect to Π , if $\tau(F) \subseteq \Delta_s^\Pi$ and $\forall F' \supset F : \tau(F') \not\subseteq \Delta_s^\Pi$. By $\mathcal{DF}(\Pi) \subset 2^{\mathcal{F}(\Pi)}$ we denote the set of delimitations over $\mathcal{F}(\Pi)$.

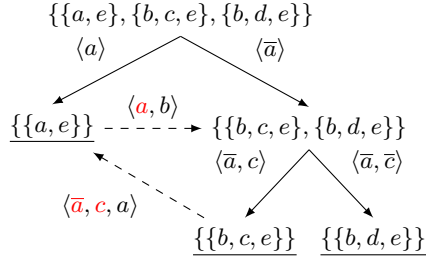


Figure 1: Goal-oriented and free navigation on program Π_1 .

We call a route consisting of delimiting facets *maximal safe*.

Definition 3. Let Π be a program, $F \subseteq \mathcal{F}(\Pi)$ and $\delta \in \tau(F) \subseteq \Delta^\Pi$. We call δ maximal safe, if and only if $F \in \mathcal{DF}(\Pi)$. By Δ_{ms}^Π we denote the set of maximal safe routes in $\mathcal{AS}(\Pi)$.

In fact, each delimitation corresponds to a unique solution.

Lemma 1. Let Π be a program, $F \subseteq \mathcal{F}(\Pi)$ and $\delta \in \tau(F) \subseteq \Delta^\Pi$. If $\delta \in \Delta_{ms}^\Pi$, then $|\mathcal{AS}(\Pi^\delta)| = 1$.

Proof. Let Π be a program, $F, F' \subseteq \mathcal{F}(\Pi)$ and $\delta \in \tau(F) \subseteq \Delta^\Pi$. Suppose $\delta \in \Delta_{ms}^\Pi$. Then $F \in \mathcal{DF}(\Pi)$ so that $\tau(F) \subseteq \Delta_s^\Pi$ and $\forall F' \supset F : \tau(F') \not\subseteq \Delta_s^\Pi$. Since $\tau(F) \subseteq \Delta_s^\Pi$, we have that $|\mathcal{AS}(\Pi^\delta)| > 0$. Note that $\mathcal{F}(\Pi^\delta) \subseteq \mathcal{F}(\Pi)$. By assumption we have $\forall F' \supset F : \tau(F') \not\subseteq \Delta_s^\Pi$, hence there is no facet $f \in \mathcal{F}(\Pi) \setminus F$ that can be activated in a way that Π^δ would not become unsatisfiable, so that $\mathcal{F}(\Pi^\delta) = \emptyset$. Now suppose $|\mathcal{AS}(\Pi^\delta)| > 1$. Then $|\mathcal{F}(\Pi^\delta)| = |\mathcal{BC}(\Pi) \setminus \mathcal{CC}(\Pi)| > 0$, which contradicts $\mathcal{F}(\Pi^\delta) = \emptyset$ and concludes the proof. \square

Theorem 1. $|\mathcal{AS}(\Pi)| = |\mathcal{DF}(\Pi)|$.

Proof. Let Π be a program and $F, F' \subseteq \mathcal{F}(\Pi)$. We need to show that $g : \mathcal{DF}(\Pi) \rightarrow \mathcal{AS}(\Pi)$ defined by $g(F) := \bigcup \mathcal{AS}(\Pi^\delta)$ such that $\delta \in \tau(F)$ is bijective. Note that g is a total function, since by Definition 3 we have $\delta \in \Delta_{ms}^\Pi$ and due to Lemma 1, if $\delta \in \Delta_{ms}^\Pi$, then $|\mathcal{AS}(\Pi^\delta)| = 1$, so that $g(F) = \bigcup \mathcal{AS}(\Pi^\delta) \in \mathcal{AS}(\Pi)$.

Injectivity: Let $F, F' \in \mathcal{DF}(\Pi)$, $\delta \in \tau(F)$, $\delta' \in \tau(F')$ and $X, X' \subseteq \mathcal{BC}(\Pi)$. Suppose $F \neq F'$. It is easy to see that answer sets delimited by F, F' respectively are of the form $\bigcup \mathcal{AS}(\Pi^\delta) = X \cup \mathcal{CC}(\Pi)$ and $\bigcup \mathcal{AS}(\Pi^{\delta'}) = X' \cup \mathcal{CC}(\Pi)$ such that $\forall f \in F : X \models f$ and $\forall f' \in F' : X' \models f'$. However, since by assumption $F, F' \in \mathcal{DF}(\Pi)$ and $F \neq F'$, there exists a facet $f'' \in F \cup F'$ that is not satisfied by both X and X' , hence $X \neq X'$, so that $\bigcup \mathcal{AS}(\Pi^\delta) \neq \bigcup \mathcal{AS}(\Pi^{\delta'})$. Therefore by contraposition, if $g(F) = g(F')$, then $F = F'$.

Surjectivity: We need to show that $\forall X \in \mathcal{AS}(\Pi) \exists F \in \mathcal{DF}(\Pi) : g(F) = X$. Let $X \in \mathcal{AS}(\Pi)$ and $F' \subseteq \mathcal{F}^+(\Pi) \subseteq \mathcal{BC}(\Pi)$ be an arbitrary set of inclusive facets of Π . Note that, since $F' \subseteq \mathcal{BC}(\Pi)$, we can characterize any answer set $X \in \mathcal{AS}(\Pi)$ by $X = F' \cup \mathcal{CC}(\Pi)$. We can make the distinction of cases:

1. Suppose $F' \neq \emptyset$. Then, since $F' \subseteq \mathcal{F}^+(\Pi)$, there exists at least one route $\delta' \in \tau(F') \subseteq \Delta^\Pi$ such that $\bigcup \mathcal{AS}(\Pi^{\delta'}) = X = F' \cup \mathcal{CC}(\Pi)$. It is easy to see that we can extend F' to F'' by adding all facets $\bar{\alpha} \in \mathcal{F}^-(\Pi)$ such that $\alpha \notin F'$, thus $X \models \bar{\alpha}$, in order to obtain a maximal safe route $\delta'' \in \tau(F'') \subseteq \Delta_{ms}^\Pi$, which points to X . Therefore $g(F'') = X$.
2. Suppose $F' = \emptyset$. Then $X = \mathcal{CC}(\Pi)$. Note that $\forall \alpha \in \mathcal{F}(\Pi) : \emptyset \models \bar{\alpha}$ and $\emptyset \not\models \alpha$. Therefore routes to reach X by must contain at least all exclusive facets $f \in \mathcal{F}^-(\Pi)$ and no inclusive facets $f' \in \mathcal{F}^+(\Pi)$ of Π , hence we can conclude that if $\delta \in \tau(\mathcal{F}^-(\Pi))$, then $\bigcup \mathcal{AS}(\Pi^\delta) = X$. It is easy to see that if a supersequence δ' of δ contains no inclusive facet, then δ' is equivalent to δ , and otherwise δ' is not safe. Therefore δ has to be maximal safe and $\mathcal{F}^-(\Pi)$ has to be delimiting, hence $g(\mathcal{F}^-(\Pi)) = X$.

Since g is a bijection, we conclude $|\mathcal{AS}(\Pi)| = |\mathcal{DF}(\Pi)|$. □

As mentioned, using routes and facets, there are several ways to explore solutions. A *navigation mode* is a function that prunes the solution space according to a search strategy that involves routes and facets.

Definition 4. Let $X_i \in 2^{\Delta^\Pi} \cup 2^{\mathcal{F}(\Pi)}$ where $0 \leq i \leq n \in \mathbb{N}$. A navigation mode is a function

$$\nu : X_0 \times \cdots \times X_n \rightarrow 2^{\mathcal{AS}(\Pi)}$$

that maps an n -ary Cartesian product over subsets of routes over Π and facets of Π to answer sets of Π .

The idea of *free* and *goal-oriented* navigation was mentioned by Alrabbaa *et al.* (2018). While free navigation follows no particular strategy, during goal-oriented navigation, we narrow down the solution space. Next, we formalize the goal-oriented navigation mode.

Definition 5. We define the goal-oriented navigation mode $\nu_{go} : \Delta_s^\Pi \times \mathcal{F}(\Pi) \rightarrow 2^{\mathcal{AS}(\Pi)}$ by:

$$\nu_{go}(\delta, f) := \begin{cases} \mathcal{AS}(\Pi^{\langle \delta, f \rangle}), & \text{if } f \in \mathcal{F}(\Pi^\delta); \\ \mathcal{AS}(\Pi^\delta), & \text{otherwise.} \end{cases}$$

As illustrated in Figure 1, while during goal-oriented navigation (indicated by solid lines) the space is being narrowed down, until some unique solution (indicated by underscores) is found, in free mode (indicated by both dashed and solid lines) unsafe routes are being redirected, as illustrated on route $\langle a, b \rangle$ where a is retracted. We call the effect of narrowing down the space *zooming in*, the inverse effect *zooming out* and any effect where the number of solutions remains the same, *slide* effect, e.g., activating a on route $\langle \bar{a}, c \rangle$.

Weighted Faceted Navigation

During faceted navigation, we can zoom in, zoom out or slide. However, we are unaware of how big the effect of activating a facet will be. Recall that different routes can lead to the same unique solution. The activation of some facet may lead to a unique solution more quickly or less quickly than the activation of another facet, which means that during navigation one has no information on the length of a route. Our framework provides an approach for consciously zooming in on solutions. Introducing *weighted* navigation, we characterize a navigation step with respect to the extent to which it affects the size of the solution space, thereby we can navigate toward solutions at a configurable “pace” of navigation, which we consider to be the extent to which the current route zooms into the solution space.

The kind of parameter that allows for configuration is called the *weight* of a facet. Weights of facets enable users to inspect effects of facets at any stage of navigation, which allows for navigating more interactively in a systematic way. Any weight or pace is associated with a *weighting function* that can be defined in various ways, specifying the number of program-related objects, e.g., answer sets.

Definition 6. Let Π be a program, $\delta \in \Delta^\Pi$, $f \in \mathcal{F}(\Pi)$ and $\delta' \in \mathcal{R}(\delta, f)$. We call $\# : \{\Pi^\delta \mid \delta \in \Delta^\Pi\} \rightarrow \mathbb{N}$ a weighting function, whenever $\#(\Pi^\delta) > 0$, if $|\mathcal{AS}(\Pi)| \geq 2$. The weight $\omega_\#$ of f with respect to $\#, \Pi^\delta$ and δ' is defined as:

$$\omega_\#(f, \Pi^\delta, \delta') := \begin{cases} \#(\Pi^\delta) - \#(\Pi^{\delta'}), & \text{if } \langle \delta, f \rangle \notin \Delta_s^{\Pi^\delta} \text{ and } \delta' \neq \epsilon; \\ \#(\Pi^\delta) - \#(\Pi^{\langle \delta, f \rangle}), & \text{otherwise.} \end{cases}$$

The pace indicates the zoom-in effect of a route with respect to a weighting function.

Definition 7. Let Π be a program such that $|\mathcal{AS}(\Pi)| \geq 2$ and $\delta \in \Delta_s^\Pi$. We define the pace $\mathcal{P}_\#(\delta)$ of δ with respect to $\#$ as $\mathcal{P}_\#(\delta) := \frac{\#(\Pi) - \#(\Pi^\delta)}{\#(\Pi)}$.

Before we instantiate weights with actual weighting functions, we identify desirable properties of weights. Most importantly, weights should indicate zoom-in effects of facets on safe routes, i.e., a weight should identify which facets lead to a proper sub-space of answer sets.

Definition 8. We call a weight $\omega_\#$ safe-zooming, whenever if $f \in \mathcal{F}(\Pi^\delta)$, then $\omega_\#(f, \Pi^\delta, \epsilon) > 0$ for $\delta \in \Delta_s^\Pi$.

Essentially, whenever a weight is *safe-zooming* it is useful to inspect zoom-in effects during goal-oriented navigation.

Definition 9. We call a weight $\omega_\#$ splitting, if $\#(\Pi^\delta) = \omega_\#(\alpha, \Pi^\delta, \delta') + \omega_\#(\bar{\alpha}, \Pi^\delta, \delta')$ for $\delta, \delta' \in \Delta_s^\Pi$ and $\alpha, \bar{\alpha} \in \mathcal{F}(\Pi^\delta)$.

Splitting weights are useful during goal-oriented navigation, as any permissible route δ in ν_{g_o} is safe and if $\#(\Pi^\delta)$ and the weight of a facet $f \in \mathcal{F}(\Pi^\delta)$ for $\delta \in \Delta_s^\Pi$ are known, we can compute the weight of the respective inverse facet $f' \in \mathcal{F}(\Pi^\delta)$ arithmetically and thus avoid computing $\#(\Pi^{\langle \delta, f' \rangle})$.

Definition 10. We call a weight $\omega_\#$ reliable, whenever $\omega_\#(f, \Pi^\delta, \epsilon) = \#(\Pi^\delta)$ if and only if $\langle \delta, f \rangle \notin \Delta_s^\Pi$ for $\delta \in \Delta^\Pi$ and $f \in \mathcal{F}(\Pi)$.

The benefit of *reliable* weights, on the other hand, is that they indicate unsafe routes. Hence, reliability can be ignored during goal-oriented navigation, but appears to be useful during free navigation.

As we are focused on narrowing down the solution space, we want to know, whether the associated weighting function $\#$ of a weight detects maximal or minimal, respectively, zoom-in effects on safe routes.

Definition 11. For a program Π , $\delta \in \Delta^\Pi$ and $f \in F$, then:

- f is maximal weighted, denoted by $f \in \max_{\omega_\#}(\Pi^\delta)$, if $\forall f' \in \mathcal{F}(\Pi^\delta) : \omega_\#(f, \Pi^\delta, \epsilon) \geq \omega_\#(f', \Pi^\delta, \epsilon)$;
- f is minimal weighted, denoted by $f \in \min_{\omega_\#}(\Pi^\delta)$, if $\forall f' \in \mathcal{F}(\Pi^\delta) : \omega_\#(f, \Pi^\delta, \epsilon) \leq \omega_\#(f', \Pi^\delta, \epsilon)$.

A weight is min-inline, if every minimal weighted facet leads to a maximal sub-space of solutions. Analogously, a weight is max-inline, if every maximal weighted facet leads to a minimal sub-space.

Definition 12. Let Π be a program, $\delta \in \Delta_s^\Pi$ and $f \in \mathcal{F}(\Pi^\delta)$. We call a weight $\omega_\#$

- min-inline, whenever $f \in \min_{\omega_\#}(\Pi^\delta)$ if and only if

$$\forall f' \in \mathcal{F}(\Pi^\delta) \setminus \min_{\omega_\#}(\Pi^\delta) : |\mathcal{AS}(\Pi^{\langle \delta, f \rangle})| > |\mathcal{AS}(\Pi^{\langle \delta, f' \rangle})|;$$

- max-inline, whenever $f \in \max_{\omega_\#}(\Pi^\delta)$ if and only if

$$\forall f' \in \mathcal{F}(\Pi^\delta) \setminus \max_{\omega_\#}(\Pi^\delta) : |\mathcal{AS}(\Pi^{\langle \delta, f \rangle})| < |\mathcal{AS}(\Pi^{\langle \delta, f' \rangle})|.$$

Below, we introduce the *absolute* weight of a facet, which counts answer sets, and two so called *relative* weights, which seek for approximating the number of solutions to compare sub-spaces with respect to their actual size, while avoiding counting.

Absolute Weight

The most natural weighting function to identify the effect of a navigation step is to observe the number of answer sets on a route. The absolute weight of a facet f is defined as the number of solutions by which the solution space grows or shrinks due to the activation of f .

Definition 13. The absolute weight $\omega_{\#\mathcal{AS}}$ is defined by $\#\mathcal{AS} : \Pi^\delta \mapsto |\mathcal{AS}(\Pi^\delta)|$.

Example 3. Let us inspect Figure 1 and the program Π_1 from Example 2. As stated by $\omega_{\#\mathcal{AS}}(a, \Pi_1^{\langle \bar{a}, c \rangle}, \langle a \rangle) = 0$, activating a on $\langle \bar{a}, c \rangle$ induces a slide. $\omega_{\#\mathcal{AS}}(a, \Pi_1^{\langle a \rangle}, \langle b \rangle) = -1$. This tells us that navigating towards b on $\langle a \rangle$ zooms out by one solution. In contrast, $\omega_{\#\mathcal{AS}}(b, \Pi_1^{\langle c \rangle}, \langle \bar{a} \rangle) = 1$ means that we zoom in by one solution.

By definition, the absolute weight directly reflects the effect of a navigation step and satisfies all introduced properties.

Theorem 2. The absolute weight $\omega_{\#\mathcal{AS}}$ is safe-zooming, splitting, reliable, min-inline, and max-inline.

Proof. Let Π be a program.

safe-zooming: Follows per definition of facets and the fact that if $f \in \mathcal{F}(\Pi)$, then $\mathcal{AS}(\Pi^{\langle f \rangle}) = \{X \in \mathcal{AS}(\Pi) \mid X \models f\}$.

reliable: Let $\delta \in \Delta^\Pi$ and $f \in \mathcal{F}(\Pi^\delta)$. By Definition 13:

$$\omega_{\#\mathcal{AS}}(f, \Pi^\delta, \epsilon) = |\mathcal{AS}(\Pi^\delta)| - |\mathcal{AS}(\Pi^{\langle \delta, f \rangle})| \quad (1)$$

(\Rightarrow) Suppose $\omega_{\#\mathcal{AS}}(f, \Pi^\delta, \epsilon) = |\mathcal{AS}(\Pi^\delta)|$. Using (1) it follows that $|\mathcal{AS}(\Pi^{\langle \delta, f \rangle})| = 0$, therefore $\langle \delta, f \rangle \notin \Delta_s^\Pi$.

(\Leftarrow) Suppose $\langle \delta, f \rangle \notin \Delta_s^\Pi$. By assumption $\mathcal{AS}(\Pi^{\langle \delta, f \rangle}) = \emptyset$, so that $|\mathcal{AS}(\Pi^{\langle \delta, f \rangle})| = 0$. Therefore due to (1), we conclude that $\omega_{\#\mathcal{AS}}(f, \Pi^\delta, \epsilon) = |\mathcal{AS}(\Pi^\delta)|$.

splitting: Let $\delta, \delta' \in \Delta_s^\Pi$ and $\alpha, \bar{\alpha} \in \mathcal{F}(\Pi^\delta)$. Then, since if $f \in \{\alpha, \bar{\alpha}\} \subseteq \mathcal{F}(\Pi^\delta)$, then $\mathcal{AS}(\Pi^{\langle f \rangle}) \neq \emptyset$, it follows that $\Pi^{\langle \delta, \alpha \rangle}$ and $\Pi^{\langle \delta, \bar{\alpha} \rangle}$ are satisfiable, which means that $\langle \delta, \alpha \rangle, \langle \delta, \bar{\alpha} \rangle \in \Delta_s^{\Pi^\delta}$. Thus Definition 13 gives (1) for $f \in \{\alpha, \bar{\alpha}\}$, respectively, so that δ' can be ignored. Define $S^{\models_{\Pi^\delta} \alpha} = \{X \in \mathcal{AS}(\Pi^\delta) \mid X \models \alpha\}$ and $S^{\models_{\Pi^\delta} \bar{\alpha}} = \{X \in \mathcal{AS}(\Pi^\delta) \mid X \models \bar{\alpha}\}$. We know that $S^{\models_{\Pi^\delta} \alpha} = \mathcal{AS}(\Pi^{\langle \delta, \alpha \rangle})$ and $S^{\models_{\Pi^\delta} \bar{\alpha}} = \mathcal{AS}(\Pi^{\langle \delta, \bar{\alpha} \rangle})$. It is easy to see that

$$S^{\models_{\Pi^\delta} \alpha} \text{ and } S^{\models_{\Pi^\delta} \bar{\alpha}} \text{ form a partition of } \mathcal{AS}(\Pi^\delta) \quad (2)$$

hence:

$$\begin{aligned} |\mathcal{AS}(\Pi^\delta)| &= |S^{\models_{\Pi^\delta} \alpha}| + |S^{\models_{\Pi^\delta} \bar{\alpha}}| \\ &= |\mathcal{AS}(\Pi^{\langle \delta, \alpha \rangle})| + |\mathcal{AS}(\Pi^{\langle \delta, \bar{\alpha} \rangle})| \\ &= (|\mathcal{AS}(\Pi^\delta)| - |\mathcal{AS}(\Pi^{\langle \delta, \bar{\alpha} \rangle})|) + (|\mathcal{AS}(\Pi^\delta)| - |\mathcal{AS}(\Pi^{\langle \delta, \alpha \rangle})|) \\ &= \omega_{\#\mathcal{AS}}(\bar{\alpha}, \Pi^\delta, \delta') + \omega_{\#\mathcal{AS}}(\alpha, \Pi^\delta, \delta') \\ &= \omega_{\#\mathcal{AS}}(\alpha, \Pi^\delta, \delta') + \omega_{\#\mathcal{AS}}(\bar{\alpha}, \Pi^\delta, \delta') \end{aligned} \quad (2)$$

min-inline: Follows directly from Definition 13.

max-inline: Follows directly from Definition 13. □

Unfortunately, computing absolute weights is expensive.

Lemma 2. *Outputting the absolute weight $\omega_{\#\mathcal{AS}}$ for a given program Π and route δ is $\#\cdot\text{coNP}$ -complete.*

Proof. Membership and hardness can be easily established by the complexity of counting the number of answer sets of a disjunctive program Π , which is known to be $\#\cdot\text{coNP}$ -complete Fichte *et al.* (2017). □

Relative Weights

Since computing absolute weights is computationally expensive (Lemma 2), we aim for less expensive methods that still retain the ability to compare sub-spaces with respect to their size. Therefore, we investigate two *relative weights*.

Facet Counting. One approach to manipulating the number of solutions and to keeping track of how the number changes over the course of navigation, is to count facets.

Definition 14. *The facet-counting weight $\omega_{\#\mathcal{F}}$ is defined by $\#\mathcal{F} : \Pi^\delta \mapsto |\mathcal{F}(\Pi^\delta)|$.*

Next, we establish a positive result in terms of complexity. Therefore, recall that $\Delta_3^P \subseteq \text{PH} \subseteq \text{P}^{\#\text{P}}$ Stockmeyer (1976); Toda (1991).

Lemma 3. *Outputting the facet-counting weight $\omega_{\#\mathcal{F}}$ for a given program Π and route δ is in Δ_3^P .*

Proof. In fact, we obtain the membership result by the following construction. We have $|\mathcal{F}(\Pi^\delta)| = |\mathcal{BC}(\Pi^\delta) \setminus \mathcal{CC}(\Pi^\delta)| = |\mathcal{BC}(\Pi^\delta)| - |\mathcal{CC}(\Pi^\delta)|$. The value of $|\mathcal{BC}(\Pi^\delta)|$ is at most $|\mathcal{A}(\Pi^\delta)^+|$ and we can compute $\mathcal{BC}(\Pi^\delta)$ by checking for every atom $\alpha \in \mathcal{A}(\Pi^\delta)^+$ whether α is a brave consequence of Π^δ , which is Σ_2^P -complete Eiter and Gottlob (1995). Similar, we can check for $|\mathcal{CC}(\Pi^\delta)|$ whether $a \in \mathcal{A}(\Pi^\delta)^+$ is a cautious consequence of Π^δ , which is Π_2^P -complete Eiter and Gottlob (1995). Computing the difference of the two integers takes time $\Theta(\log n)$. □

Hence, assuming standard theoretical assumptions, counting facets is easier than counting solutions. However, below we show that counting facets has deficiencies, when it comes to comprehending the solution space regarding its size.

Lemma 4. *$|\mathcal{AS}(\Pi)| \leq 1$ if and only if $|\mathcal{F}(\Pi)| = 0$.*

Proof. Let Π be a program.

(\Rightarrow) Suppose $|\mathcal{F}(\Pi)| > 0$. Then $\mathcal{BC}(\Pi) \neq \emptyset$, so that $|\mathcal{AS}(\Pi)| > 0$. Now, suppose $|\mathcal{AS}(\Pi)| = 1$. Then $\mathcal{BC}(\Pi) = \mathcal{CC}(\Pi)$, which means that $|\mathcal{F}(\Pi)| = 0$ and contradicts $|\mathcal{F}(\Pi)| > 0$. Therefore $|\mathcal{AS}(\Pi)| > 1$, which by contraposition concludes the proposition.

(\Leftarrow) Suppose $|\mathcal{F}(\Pi)| = 0$. Then $\mathcal{BC}(\Pi) = \mathcal{CC}(\Pi)$. Due to the minimality of answer sets we conclude that therefore either $\mathcal{AS}(\Pi) = \emptyset$, so that $|\mathcal{AS}(\Pi)| = 0$, or $|\mathcal{AS}(\Pi)| = 1$. Therefore $|\mathcal{AS}(\Pi)| \leq 1$. □

From Lemma 4 and the fact that for program Π_1 from Example 2 we have $\omega_{\#\mathcal{F}}(c, \Pi_1^{\langle \bar{a} \rangle}, \epsilon) = |\mathcal{F}(\Pi_1^{\langle \bar{a} \rangle})|$, but $\langle \bar{a}, c \rangle \in \Delta_s^{\Pi_1}$, we conclude that $\omega_{\#\mathcal{F}}$ is not reliable. Furthermore, since therefore $\omega_{\#\mathcal{F}}(c, \Pi_1^{\langle \bar{a} \rangle}, \epsilon) + \omega_{\#\mathcal{F}}(\bar{c}, \Pi_1^{\langle \bar{a} \rangle}, \epsilon) \neq |\mathcal{F}(\Pi_1^{\langle \bar{a} \rangle})|$, $\omega_{\#\mathcal{F}}$ is not splitting either.

Corollary 1. *The facet-counting weight $\omega_{\#\mathcal{F}}$ is not reliable and not splitting.*

The reason for $\omega_{\#\mathcal{F}}$ not distinguishing between one and no solution is that we can interpret it as an indicator for how the diversity or similarity, respectively, of solutions changes by activating a facet. Accordingly, whenever a step leads to one or no solution, the thereby reached sub-space contains least-diverse or most-similar solutions, respectively.

Example 4. *Again consider Π_1 from Example 2. While on the absolute level $\omega_{\#\mathcal{AS}}(\bar{a}, \Pi_1, \epsilon) = 1 = \omega_{\#\mathcal{AS}}(\bar{c}, \Pi_1, \epsilon)$, counting facets, $\omega_{\#\mathcal{F}}(\bar{a}, \Pi_1, \epsilon) = 4$ and $\omega_{\#\mathcal{F}}(\bar{c}, \Pi_1, \epsilon) = 2$, the relative weights of \bar{c} and \bar{a} differ. The reason is that even though $|\mathcal{AS}(\Pi_1^{(\bar{a})})| = |\mathcal{AS}(\Pi_1^{(\bar{c})})|$, by activating \bar{c} we can still navigate towards $\mathcal{F}(\Pi_1^{(\bar{c})}) = \{a, \bar{a}, b, \bar{b}, d, \bar{d}\}$, but activating \bar{a} , we can only navigate toward $\mathcal{F}(\Pi_1^{(\bar{a})}) = \{c, \bar{c}, d, \bar{d}\}$, i.e., answer sets that contain b .*

In other words, while $\#\mathcal{F}$ indicates how “far apart” solutions are, $\omega_{\#\mathcal{F}}$ indicates to what amount the solutions converge due to navigation steps.

Theorem 3. *The facet-counting weight $\omega_{\#\mathcal{F}}$ is safe-zooming.*

Proof. Let Π be a program and $\delta \in \Delta_s^\Pi$. By Definition 14:

$$\omega_{\#\mathcal{F}}(f, \Pi^\delta, \epsilon) = |\mathcal{F}(\Pi^\delta)| - |\mathcal{F}(\Pi^{(\delta, f)})| \quad (3)$$

Suppose $f \in \{\alpha, \bar{\alpha}\} \subseteq \mathcal{F}(\Pi^\delta)$. Then we know that $\mathcal{AS}(\Pi^{(\delta, f)}) = \{X \in \mathcal{AS}(\Pi^\delta) \mid X \models f\}$, so that either $\forall X \in \mathcal{AS}(\Pi^{(\delta, f)}) : \alpha \in X$, or $\forall X \in \mathcal{AS}(\Pi^{(\delta, f)}) : \alpha \notin X$. Therefore either $\alpha \in \bigcap \mathcal{AS}(\Pi^{(\delta, f)}) = \mathcal{CC}(\Pi^{(\delta, f)})$, or $\alpha \notin \bigcup \mathcal{AS}(\Pi^{(\delta, f)}) = \mathcal{BC}(\Pi^{(\delta, f)})$. Per definition of facets in both cases $\mathcal{F}(\Pi^{(\delta, f)}) \subseteq \mathcal{F}(\Pi^\delta) \setminus \{f\}$. Therefore $|\mathcal{F}(\Pi^{(\delta, f)})| < |\mathcal{F}(\Pi^\delta)|$. Using (3) gives $\omega_{\#\mathcal{F}}(f, \Pi^\delta, \epsilon) > 0$, which concludes the proof. \square

Due to Theorem 3, we know that $\#\mathcal{F}$ can be used to determine the pace of safe navigation. In fact the facet-counting pace $\mathcal{P}_{\#\mathcal{F}}$ emphasizes that $\omega_{\#\mathcal{F}}$ is not directly related to the size of the solution space.

Example 5. *Consider Π_1 from Example 2. While $|\mathcal{AS}(\Pi_1^{(\bar{c})})| = 2$ and $|\mathcal{AS}(\Pi_1)| = 3$, which means that activating \bar{c} on Π_1 we lose 1 of 3 solutions so that $\mathcal{P}_{\#\mathcal{AS}}(\bar{c}) = \frac{1}{3}$, we have $\mathcal{P}_{\#\mathcal{F}}(\bar{c}) = \frac{1}{4}$.*

From Lemma 4, we immediately conclude:

Corollary 2. *$\mathcal{P}_{\omega_{\#\mathcal{F}}}(\delta) = 1$ if and only if $\delta \in \Delta_{ms}^\Pi$. In contrast, for all $\delta \in \Delta_s^\Pi$ we have*

$$\mathcal{P}_{\#\mathcal{AS}}(\delta) \leq \frac{|\mathcal{AS}(\Pi)| - 1}{|\mathcal{AS}(\Pi)|}.$$

Corollary 2 states that, in contrast to $\mathcal{P}_{\#\mathcal{AS}}$, the facet counting pace $\mathcal{P}_{\omega_{\#\mathcal{F}}}$ detects whether users sit on a unique solution. More importantly it is the better option to find a viable implementation of the pace of navigation for our framework. While in that sense using the relative weight $\omega_{\#\mathcal{F}}$ is beneficial, unfortunately it is not *min-inline*.

Example 6. *We consider $\Pi_2 = \{a \mid b \mid c; d \mid e \leftarrow b; f \leftarrow c\}$ where $\mathcal{AS}(\Pi_2) = \{\{a\}, \{b, d\}, \{b, e\}, \{c, f\}\}$. While $\bar{a} \in \min_{\omega_{\#\mathcal{F}}}(\Pi_2)$ and $\bar{c} \notin \min_{\omega_{\#\mathcal{F}}}(\Pi_2)$, we have $|\mathcal{AS}(\Pi_2^{(\bar{a})})| = |\mathcal{AS}(\Pi_2^{(\bar{c})})|$. Hence, the relative weight $\omega_{\#\mathcal{F}}$ is not *min-inline*.*

We suspect that the property *max-inline* is not satisfied by the weight $\omega_{\#\mathcal{F}}$ as we observed in our experiments that the activation of some facets, which had no maximal $\omega_{\#\mathcal{F}}$ weight, lead to smaller answer set spaces than the activation of facets which had maximal $\omega_{\#\mathcal{F}}$ weight. An actual counterexample is still open.

Supported Model Counting. Another approach to comparing sub-spaces with respect to their size, while avoiding answer set counting, is to count supported models. An interpretation X is called *supported model* Apt *et al.* (1988); Alviano and Dodaro (2016b) of Π if X satisfies Π and for all $\alpha \in X$ there is a rule $r \in \Pi$ such that $H(r) \cap X = \{\alpha\}$, $B^+(r) \subseteq X$ and $B^-(r) \cap X = \emptyset$. By $\mathcal{S}(\Pi)$ we denote the supported models of Π . It holds that $\mathcal{AS}(\Pi) \subseteq \mathcal{S}(\Pi)$ Marek and Subrahmanian (1992), but the converse does not hold in general. We define *supp weights*, by which in short we refer to supported model counting weights, accordingly as follows.

Definition 15. The *supp weight* $\omega_{\#S}$ is defined by $\#S : \Pi^\delta \mapsto |\mathcal{S}(\Pi)|$.

The *positive dependency graph* of program Π is $G(\Pi) := (\mathcal{A}(\Pi), \{(\alpha_1, \alpha_0) \mid \alpha_1 \in B^+(r), \alpha_0 \in H(r), r \in \Pi\})$. Π is called *tight*, if $G(\Pi)$ is acyclic. If Π is tight, then models of the completion and answer sets coincide Fages (1994).

Since we have $\mathcal{AS}(\Pi) = \mathcal{S}(\Pi)$ for tight programs Π , we can immediately obtain the following corollary.

Corollary 3. If Π is tight, then for all $f \in \mathcal{F}(\Pi^\delta)$ we have that $\omega_{\#AS}(f, \Pi^\delta, \delta') = \omega_{\#S}(f, \Pi^\delta, \delta')$.

Due to the fact that unsatisfiable programs may have supported models Marek and Subrahmanian (1992), $\omega_{\#S}$ is not reliable. Moreover the following example shows that $\omega_{\#S}$ is neither min-inline, nor max-inline.

Example 7. We consider $\Pi_3 = \{a; b \leftarrow a, \sim c; c \leftarrow \sim b, \sim d; d \leftarrow d\}$ with $\mathcal{S}(\Pi_3) = \{\{a, b\}, \{a, c\}, \{a, b, d\}\}$ and $\mathcal{AS}(\Pi_3) = \{\{a, b\}, \{a, c\}\}$. The facets of Π_3 are given by $\mathcal{F}(\Pi_3) = \{b, \bar{b}, c, \bar{c}\}$. Then, the facets b and \bar{c} both have *supp weight* 1 and thus are *minimal weighted*, and the facets c and \bar{b} have *supp weight* 2 and thus are *maximal weighted*. As $|\mathcal{AS}(\Pi_3^{(b)})| = |\mathcal{AS}(\Pi_3^{(c)})| = 1$ we see that both the *minimal* and the *maximal weighted facets* with respect to *supp weights* have the same number of answer sets. Hence, $\omega_{\#S}$ is neither min-inline, nor max-inline.

Although $\omega_{\#S}$ does not satisfy min-inline and max-inline, it shares some properties with $\omega_{\#AS}$ and $\omega_{\#F}$.

Lemma 5. Let Π be a program and $\delta \in \Delta_s^\Pi$. If $f \in \mathcal{F}(\Pi^\delta)$, then

$$\mathcal{S}(\Pi^{(\delta, f)}) = \{X \in \mathcal{S}(\Pi^\delta) \mid X \models f\} \subset \mathcal{S}(\Pi^\delta).$$

Proof. Let Π be a program and $\delta \in \Delta_s^\Pi$. Suppose $f \in \{\alpha, \bar{\alpha}\} \subseteq \mathcal{F}(\Pi^\delta)$. Then, we know that $\mathcal{AS}(\Pi^{(\delta, f)}) \neq \emptyset$, so that, using the fact that $\mathcal{AS}(\Pi) \subseteq \mathcal{S}(\Pi)$, we conclude that $\mathcal{S}(\Pi^{(\delta, f)}) \neq \emptyset$. It is well known that an integrity constraint $\leftarrow \alpha$ can be encoded as a self-blocking rule $\alpha' \leftarrow \alpha, \sim \alpha'$ where α' is a new introduced atom, so that $ic(f)$ can be encoded as $\alpha' \leftarrow \alpha, \sim \alpha'$ ($\alpha' \leftarrow \sim \alpha, \sim \alpha'$ respectively). Hence, by definition of $\mathcal{S}(\Pi)$, it is easy to see that activating $f = \alpha$ rejects any interpretation $X \in \mathcal{S}(\Pi^\delta)$ that contains α . Analogously, if $f = \bar{\alpha}$ any interpretation that does not contain α is being rejected. Therefore we conclude that $\mathcal{S}(\Pi^{(\delta, f)}) = \{X \in \mathcal{S}(\Pi^\delta) \mid X \models f\} \subset \mathcal{S}(\Pi^\delta)$. □

Theorem 4. The *supp weight* $\omega_{\#S}$ is *safe-zooming* and *splitting*.

Proof. Let Π be a program, $\delta \in \Delta_s^\Pi$ and $f \in \mathcal{F}(\Pi^\delta)$.

safe-zooming: Follows directly from Lemma 5.

splitting: Suppose $f \in \{\alpha, \bar{\alpha}\}$. Due to Lemma 5 it is easy to see that $\mathcal{S}(\Pi^{(\delta, \alpha)})$ and $\mathcal{S}(\Pi^{(\delta, \bar{\alpha})})$ form a partition of $\mathcal{S}(\Pi^\delta)$, from which, analogously to the proof for the splitting property of $\omega_{\#AS}$, it follows that $\omega_{\#S}$ is splitting. □

Computing *supp weights* is computationally easier.

Lemma 6. *Outputting the supp weight $\omega_{\#S}$ for a given program Π and route δ is #P-complete.*

Proof. Since we can easily compute $\omega_{\#S}$ using Clark’s completion Clark (1978) and propositional model counting Valiant (1979) and vice-versa encode a SAT instance into a logic program while preserving the models Niemelä (1999), we obtain membership and hardness. □

However, recalling Lemma 3, note that counting facets is still the least expensive method.

	<i>saf</i>	<i>rel</i>	<i>spl</i>	<i>min</i>	<i>max</i>
$\omega_{\#AS}$	✓	✓	✓	✓	✓
$\omega_{\#F}$	✓	✗	✗	✗	?
$\omega_{\#S}$	✓	✗	✓	✗	✗

Table 1: Comparing weights regarding **saf**: is safe-zooming, **spl**: is splitting, **rel**: is reliable, **min**: is min-inline and **max**: is max-inline.

In summary, we can characterize and compare the introduced weights as given in Table 1. Every weight has its advantages that should be used to leverage performance, or characterize the solution space and its sub-spaces. While counting solutions is the most desirable choice, computing $\omega_{\#AS}$ is hard. Our results show that, when narrowing down the space by strictly pruning the maximum/minimum number of solutions, at least for tight programs, $\omega_{\#S}$ is the best choice, as it coincides with $\omega_{\#AS}$ while remaining less expensive. In general, in contrast to $\omega_{\#AS}$, relative weights come with different use cases regarding their interpretation. Even though $\omega_{\#F}$ has deficiencies, it satisfies the most essential property, namely being safe-zooming, and provides information on the similarity/diversity of solutions w.r.t. a route. To conclude, while facet-counting is the most promising method for distinguishing zoom-in effects of facets regarding computational feasibility, counting supported models of tight programs is precise about zoom-in effects.

Weighted Navigation Modes

In the following, we introduce two new navigation modes, called *strictly goal-oriented* and *explore*. They can be understood as special cases of goal-oriented navigation.

Definition 16. *Let Π be a program, $\delta \in \Delta_s^\Pi$ and $f \in \mathcal{F}(\Pi)$. The strictly goal-oriented mode $\nu_{sgo}^\#$ and the explore $\nu_{expl}^\#$ mode are defined by:*

$$\nu_{sgo}^\#(\delta, f) := \begin{cases} \mathcal{AS}(\Pi^{(\delta, f)}), & \text{if } f \in \max_{\omega_\#}(\Pi^\delta); \\ \mathcal{AS}(\Pi^\delta) & \text{otherwise.} \end{cases}$$

$$\nu_{expl}^\#(\delta, f) := \begin{cases} \mathcal{AS}(\Pi^{(\delta, f)}), & \text{if } f \in \min_{\omega_\#}(\Pi^\delta); \\ \mathcal{AS}(\Pi^\delta) & \text{otherwise.} \end{cases}$$

Corollary 4. *$\nu_{sgo}^\#$ and $\nu_{expl}^\#$ avoid unsafe routes, hence we can use the restriction $\omega_\#|_X$ of $\omega_\#$ where $X := \{(f, \delta, \epsilon) \mid f \in \mathcal{F}(\Pi), \delta \in \Delta_s^\Pi\}$.*

While in strictly goal-oriented mode the objective is to “rush” through the solution space, navigating at the highest possible pace in order to reach a unique solution as quick as possible, explore mode keeps the user off one unique solution as long as possible, aiming to provide her with as many solutions as possible to explore while “strolling” between sub-spaces. As a consequence, regardless of whether absolute or relative weights are used, during weighted navigation some (partial) solutions may be unreachable.

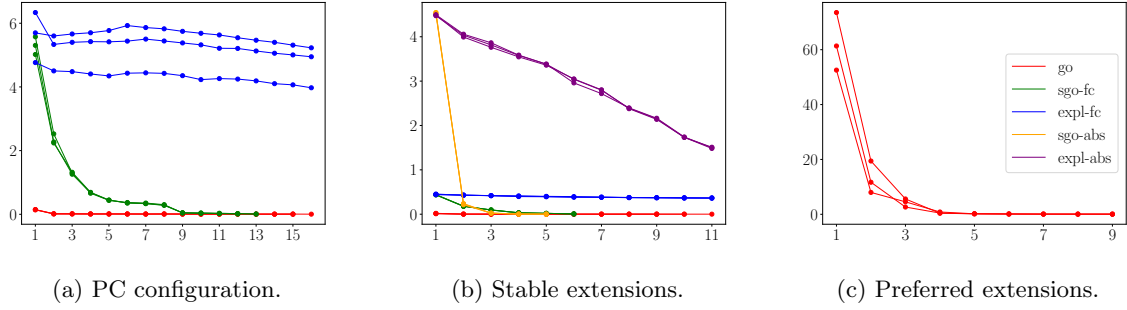


Figure 2: Comparing random steps in several navigation modes. The x-axis refers to the respective navigation step, the y-axis refers to the execution time in seconds. Colors in Figure 2a and 2b follow the legend as given in Figure 2c.

Example 8. Consider Π_2 from Example 6 where we can choose from facets $\mathcal{F}(\Pi_2) = \{a, b, c, d, e, f, \bar{a}, \bar{b}, \bar{c}, \bar{d}, \bar{e}, \bar{f}\}$ and $\max_{\omega_{\#AS}}(\Pi_2) = \{a, c, d, e, f\} = \max_{\omega_{\#F}}(\Pi_2)$. Thus, any solution $X \in AS(\Pi_2) = \{\{a\}, \{b, d\}, \{b, e\}, \{c, f\}\}$ such that $b \in X$ is unreachable in $\nu_{sgo}^{\#AS}$ and $\nu_{sgo}^{\#F}$. Accordingly, since $\omega_{\#AS}$ is splitting, it follows that $\min_{\#AS}(\Pi_2) = \{\bar{a}, \bar{c}, \bar{d}, \bar{e}, \bar{f}\}$. Hence, navigating in $\nu_{expl}^{\#AS}$, one has to sacrifice either partial solution a , or c and f right in the beginning. Furthermore, since $\min_{\omega_{\#F}}(\Pi_2) = \{\bar{a}, \bar{d}, \bar{e}\}$, right in the beginning of navigating in $\nu_{expl}^{\#F}$, one has to sacrifice partial solution a , d , or e .

Implementation and Evaluation

To study the feasibility of our framework, we implemented the *faceted answer set browser* (**fasb**) on top of the **clingo** solver. In particular, we conducted experiments on three instance sets that range from large solution spaces to complex encodings in order to verify the following two hypotheses: **(H1)** weighted faceted navigation can be performed in reasonable time in an incomprehensible solution space associated with product configuration; and **(H2)** the feasibility of our framework depends on the complexity of the given problem, i.e., program. The implementation and experiments are publicly available Fichte *et al.* (2021d,e).

Environment. **fasb** is designed for desktop systems, enabling users to practicably explore the solution space in an interactive way. Hence, runtime was limited to 600 seconds and the experiments were run on an eight core Intel i7-10510U CPU 1.8 GHz with 16 GB of RAM, running Manjaro Linux 21.1.1 (kernel 5.10.59-1-MANJARO). Runtime was measured in elapsed time by timers in **fasb** itself.

Design of Experiment. Currently, we miss data on real user behavior. Thus, we run three iterations of random navigation steps in each of the implemented modes, to simulate a user and avoid bias regarding the choice of steps. For *go*, *sgo-fc*, and *sgo-abs*, we use the `--random-safe-walk` call, which in the provided mode performs random steps until the current route is maximal safe, e.g., in *sgo-fc* and *sgo-abs* it computes maximal weighted facets and then chooses one of them to activate randomly. Since, in practice, using *expl-fc* and *expl-abs*, we do not necessarily aim to arrive at a unique solution, we use `--random-safe-steps` for *expl-fc* and *expl-abs* and provide the maximum number n of steps among iterations in *go*, which performs n random steps in the provided mode. We measure the elapsed time for a mode to filter current facets according to its strategy, then, using the mentioned calls, we randomly select a facet thereof to activate, until we reach a unique solution or took n steps. For any mode except *go*, we ignore the elapsed time of `--activate`, for *go* we solely measure elapsed time of the `--activate` call, which in the case of *go* includes

runtime of computing facets. `fasb` computes the initial facets at startup, which are used throughout further computations, in particular when performing a first step. Thus, we add elapsed time, due to startup, to the first result in each mode.

Instances. To study (H1), we inspect product configuration Gorczyca (2020) where users may configure PC components over a large solutions space until a full configuration is obtained. To verify (H2), we select instances from *abstract argumentation* using the ASPARTIX fixed ASP encodings (Dvořák *et al.*, 2020) `stable.lp` and `preferred-cond-disj.dl`. There, brave and cautious reasoning in abstract argumentation is of higher complexity for preferred semantics than for the stable semantics Baroni *et al.* (2011). For the *stable* argumentation semantics, the problems can be encoded as normal programs. Whereas for preferred, one needs disjunctive programs. As input instance, we used the abstract argumentation framework `A/3/ferry2.pfile-L3-C1-06.pddl.1.cnf.apx` from the benchmark set of (ICCMA'17) Gaggl *et al.* (2020). There, solutions of both semantics coincide with exactly 7696 answer sets.

Observations and Results. In the beginning of PC configuration, we choose from 340 facets resulting in on average in 15 steps in *go* and 13 steps in *sgo-fc* to reach a unique solution. Taking 16 steps in *expl-fc*, throughout all iterations the facet-counting pace of the obtained route is 9%. The number of solutions for the respective generated benchmark `pc_config` remains unknown. Running `clingo` for over 9 hours resulted in more than $1.3 \cdot 10^9$ answer sets. As expected, for more than a billion solutions, *sgo-abs* and *expl-abs* timed out in the first step. Inspecting Figure 2a, we see that *sgo-fc* execution time drops significantly from Step 1 to 5, which originates in the fact that Steps 1 to 5 throughout all iterations on the average decreased the number of remaining facets by 35%. Consequently, it reduces the number of facets to compute weights for and leads to shorter execution times. In *expl-fc*, on the other hand, throughout all iterations each step decreases the facet-count by 2. Except for one outlier, this leads to slowly decreasing, but in general, similar execution times. Figures 2b and 2c illustrate the execution times for navigation steps in the argumentation instances. As expected, we see no timeouts when navigating through 7696 stable extensions. Whereas exploring 7696 preferred extensions, works only in mode *go*. Computing cautious consequences was most expensive when considering the execution time of processes at startup for preferred extensions, which emphasizes (H2). From Figure 2b, we see that *go*, *sgo-fc*, and *expl-fc* show a similar trend to Figure 2a. While *go* and *expl-fc* remain rather steady in execution time, *sgo-fc* drops in the first steps. Moreover, we observe that the execution time of *expl-abs*, in contrast to *expl-fc*, decreases noticeably with every step indicating that counting less answer sets in each step becomes easier, whereas counting facets does not. Throughout all iterations, while *sgo-fc* needs 6 steps, *sgo-abs* only needs 5 steps to reach a unique solution. The significant drop between Step 1 and 2 in *sgo-abs* originates in zooming in by 93%, pruning 7152 out of 7696 solutions.

Summary. In general (H2) the feasibility of weighted navigation depends on the complexity of the given problem. Regarding product configuration, associated with a large and incomprehensible solution space (H1), weighted navigation can be performed in reasonable time using `fasb`.

Conclusion and Future Work

We provide a formal, dynamic, and flexible framework for navigating through subsets of answer sets in a systematic way. We introduce absolute and relative weights to quantify the size of the search space when reasoning under assumptions (facets) as well as natural navigation operations. In a systematic comparison, we prove which weights can be employed under the search space navigation operations. In addition, we illustrate the computational complexity for computing the weights. Our framework is intended as an additional layer on top of a solver, adding functionality for systematically manipulating the size of the

solution space during (faceted) answer set navigation. Our implementation, on top of the solver `clingo`, demonstrates feasibility of our framework for an incomprehensible solution space.

For future work, we believe that an interesting question is to research relative weights which preserve the properties `min-inline` and `max-inline`. Furthermore, we aim to investigate whether supported model counting is in fact practically feasible using recent developments in propositional model counting Bendík and Meel (2020); Fichte *et al.* (2021a,c,b); Korhonen and Jarvisalo (2021) and ASP Fichte and Hecher (2019).

Acknowledgements

The authors are stated in alphabetic order. This research was partially funded by the DFG through the Collaborative Research Center, Grant TRR 248 see <https://perspicuous-computing.science> project ID 389792660, the Bundesministerium für Bildung und Forschung (BMBF), Grant 01IS20056_NAVAS, a Google Fellowship at the Simons Institute, and the Austrian Science Fund (FWF), Grant Y698. Work has partially been carried out while Johannes Fichte was visiting the Simons Institute for the Theory of Computing.

References

- Christian Alrabbaa, Sebastian Rudolph, and Lukas Schweizer. Faceted answer-set navigation. In Christoph Benzmüller, Francesco Ricca, Xavier Parent, and Dumitru Roman, editors, *Proc. of the 2nd Int. Joint Conf. on Rules and Reasoning (RuleML+RR'18)*, pages 211–225. Springer, 2018.
- Mario Alviano and Carmine Dodaro. Anytime answer set optimization via unsatisfiable core shrinking. *TPLP*, 16(5-6):533–551, 2016.
- Mario Alviano and Carmine Dodaro. Completion of disjunctive logic programs. In Subbarao Kambhampati, editor, *Proc. of the 25th Int. Joint Conf. on Artificial Intelligence (IJCAI'16)*, pages 886–892. IJCAI/AAAI Press, 2016.
- Mario Alviano, Carmine Dodaro, Nicola Leone, and Francesco Ricca. Advances in wasp. In Francesco Calimeri, Giovambattista Ianni, and Mirosław Truszczyński, editors, *Proc. of the 13th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'15)*, pages 40–54. Springer, 2015.
- Mario Alviano, Francesco Calimeri, Carmine Dodaro, Davide Fuscà, Nicola Leone, Simona Perri, Francesco Ricca, Pierfrancesco Veltri, and Jessica Zangari. The ASP system DLV2. In Marcello Balduccini and Tomi Janhunen, editors, *Proceedings of the 14th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'17)*, volume 10377 of *LNCS*, pages 215–221, Cham, 2017. Springer.
- Mario Alviano, Javier Romero, and Torsten Schaub. Preference relations by approximation. In Michael Thielscher and Francesca Toni, editors, *Proc. of the 16th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'18)*, pages 2–11, 2018.
- Krzysztof R Apt, Howard A Blair, and Adrian Walker. Towards a theory of declarative knowledge. In *Foundations of deductive databases and logic programming*, pages 89–148. Elsevier, 1988.
- Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
- Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *Knowledge Eng. Review*, 26:365–410, 12 2011.
- Jaroslav Bendík and Kuldeep S Meel. Approximate counting of minimal unsatisfiable subsets. In Shuvendu K. Lahiri and Chao Wang, editors, *Proceeding of the 32nd Int. Conf. on Computer Aided Verification (CAV'20)*, pages 439–462. Springer, 2020.
- Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.
- Gerhard Brewka, James Delgrande, Javier Romero, and Torsten Schaub. asprin: Customizing answer set preferences without a headache. In *Proc. of the 29th AAAI Conf. on Artificial Intelligence (AAAI'15)*, 2015.

- Gerhard Brewka, James Delgrande, Javier Romero, and Torsten Schaub. Implementing preferences with asprin. In Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczyński, editors, *Proc. of the 13th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'15)*, volume 9345, pages 158–172. Springer, 2015.
- Gerhard Brewka. Complex preferences for answer set optimization. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *Proc. of the 9th Int. Conf. on Knowledge Representation and Reasoning (KR'04)*, pages 213–223. The AAAI Press, 2004.
- Francesco Calimeri, Wolfgang Faber, Martin Gebser, Giovambattista Ianni, Roland Kaminski, Thomas Krennwallner, Nicola Leone, Marco Maratea, Francesco Ricca, and Torsten Schaub. Asp-core-2 input language format. *TPLP*, 20(2):294–309, 2020.
- Keith L Clark. Negation as failure. In *Logic and data bases*, pages 293–322. Springer, 1978.
- Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranajit B. Banerji, and Jeffrey D. Ullman, editors, *Proc. of the 3rd Annual Symposium on Theory of Computing (ACM STOC'71)*, pages 151–158. ACM, 1971.
- Yannis Dimopoulos, Bernhard Nebel, and Jana Koehler. Encoding planning problems in nonmonotonic logic programs. In Sam Steel and Rachid Alami, editors, *Proc. of the 4th European Conf. on Planning (ECP'97)*, pages 169–181. Springer, 1997.
- Carmine Dodaro, Philip Gasteiger, Kristian Reale, Francesco Ricca, and Konstantin Schekotihin. Debugging non-ground ASP programs: Technique and graphical tools. *TPLP*, 19(2):290–316, 2019.
- Arnaud Durand, Miki Hermann, and Phokion G. Kolaitis. Subtractive reductions and complete problems for counting complexity classes. *Theor. Comput. Sci.*, 340(3):496–513, 2005.
- Wolfgang Dvořák, Sarah Alice Gaggl, Anna Rapberger, Johannes Peter Wallner, and Stefan Woltran. The ASPARTIX system suite. In Henry Prakken, Stefano Bistarelli, Francesco Santini, and Carlo Taticchi, editors, *Proc. of Computational Models of Argument (COMMA'20)*, volume 326 of *FAIA*, pages 461–462. IOS Press, 2020.
- Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Proc. of the 6th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'03)*, pages 502–518. Springer, 2003.
- Thomas Eiter and Georg Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Ann. Math. Artif. Intell.*, 15(3–4):289–323, 1995.
- Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. Answer set programming: A primer. In Sergio Tessaris, Enrico Franconi, Thomas Eiter, Claudio Gutierrez, Siegfried Handschuh, Marie-Christine Rousset, and Renate A. Schmidt, editors, *Proc. of the 5th Int. Summer School (Reasoning Web'09)*, pages 40–110. Springer, 2009.
- Omar El-Khatib, Enrico Pontelli, and Tran Cao Son. Justification and debugging of answer set programs in asp. In *Proceedings of the 6th International Symposium on Automated Analysis-Driven Debugging (AADEBUG'05)*, pages 49–58. ACM, 2005.
- Flavio Everardo, Tomi Janhunen, Roland Kaminski, and Torsten Schaub. The return of xorro. In Marcello Balduccini, Yuliya Lierler, and Stefan Woltran, editors, *Proc. of the 15th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'19)*, pages 284–297. Springer, 2019.

- Flavio Everardo. Towards an automated multitrack mixing tool using answer set programming. In *14th Sound and Music Computing Conf*, 2017.
- Francois Fages. Consistency of Clark’s completion and existence of stable models. *Journal of Methods of logic in computer science*, 1(1):51–60, 1994.
- Johannes K. Fichte and Markus Hecher. Treewidth and counting projected answer sets. In Marcello Balduccini, Yuliya Lierler, and Stefan Woltran, editors, *Proceedings of the 15th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR’19)*, volume 11481 of *LNCS*, pages 105–119, Philadelphia, PA, USA, 2019. Springer.
- Johannes K. Fichte, Markus Hecher, Michael Morak, and Stefan Woltran. Answer set solving with bounded treewidth revisited. In Marcello Balduccini and Tomi Janhunen, editors, *Proc. of the 14th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR’17)*, volume 10377 of *LNCS*, pages 132–145. Springer, 2017.
- Johannes K. Fichte, Markus Hecher, and Florim Hamiti. The model counting competition 2020. *ACM Journal of Experimental Algorithmics*, 26(13), December 2021.
- Johannes K. Fichte, Markus Hecher, and Valentin Roland. Parallel model counting with CUDA: Algorithm engineering for efficient hardware utilization. In Laurent D. Michel, editor, *Proceedings of the 27th International Conference on Principles and Practice of Constraint Programming (CP’21)*, volume 210 of *LIPIcs*, pages 24:1–24:20, Dagstuhl, Germany, 2021. Dagstuhl Publishing.
- Johannes K. Fichte, Markus Hecher, Patrick Thier, and Stefan Woltran. Exploiting database management systems and treewidth for counting. *TPLP*, pages 1–30, 2021.
- Johannes K. Fichte, Sarah Alice Gaggl, and Dominik Rusovac. Rushing and Strolling among Answer Sets - Navigation Made Easy (Experiments). <https://doi.org/10.5281/zenodo.5768085>, December 2021.
- Johannes K. Fichte, Sarah Alice Gaggl, and Dominik Rusovac. Rushing and Strolling among Answer Sets - Navigation Made Easy (Faceted Answer Set Browser fasb). <https://doi.org/10.5281/zenodo.5767980>, December 2021.
- Sarah Alice Gaggl, Thomas Linsbichler, Marco Maratea, and Stefan Woltran. Design and results of the second international competition on computational models of argumentation. *Artif. Intell.*, 279, 2020.
- Martin Gebser, Jörg Pührer, Torsten Schaub, and Hans Tompits. A meta-programming technique for debugging answer-set programs. In *Proc. of the 23rd AAAI Conf. on Artificial Intelligence (AAAI’08)*, 2008.
- Martin Gebser, Roland Kaminski, Arne König, and Torsten Schaub. Advances in gringo series 3. In James P. Delgrande and Wolfgang Faber, editors, *Proc. of the Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR’11)*, pages 345–351. Springer, 2011.
- Martin Gebser, Roland Kaminski, and Torsten Schaub. Complex optimization in answer set programming. *TPLP*, 11(4-5):821–839, 2011.
- Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Answer set solving in practice. *Synthesis lectures on artificial intelligence and machine learning*, 6(3):1–238, 2012.
- Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo = ASP + control: Preliminary report. *CoRR*, abs/1405.3694, 2014.

- Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Proc. of the 5th Int. Conf. and Symposium on Logic Programming (ICLP/SLP'88)*, volume 2, pages 1070–1080. MIT Press, August 1988.
- Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.
- Piotr Gorczyca. Configuration Problem ASP Encoding Generator. <https://doi.org/10.5281/zenodo.5777217>, November 2020.
- Lane A. Hemaspaandra and Heribert Vollmer. The satanic notations: Counting classes beyond #P and other definitional adventures. *SIGACT News*, 26(1):2–13, March 1995.
- Tuukka Korhonen and Matti Järvisalo. Integrating Tree Decompositions into Decision Heuristics of Propositional Model Counters. In Laurent D. Michel, editor, *Proc. of the 27th Int. Conference on Principles and Practice of Constraint Programming (CP'21)*, volume 210 of *LIPICs*, pages 8:1–8:11. Dagstuhl Publishing, 2021.
- Joohyung Lee, Samidh Talsania, and Yi Wang. Computing lpmln using asp and mln solvers. *TPLP*, 17(5-6):942–960, 2017.
- Vladimir Lifschitz. Action languages, answer sets, and planning. In *The Logic Programming Paradigm*, pages 357–373. Springer, 1999.
- W Marek and VS Subrahmanian. The relationship between stable, supported, default and autoepistemic semantics for general logic programs. *Theor. Comput. Sci.*, 103(2):365–386, 1992.
- Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. Artif. Intell.*, 25(3-4):241–273, 1999.
- Monica Nogueira, Marcello Balduccini, Michael Gelfond, Richard Watson, and Matthew Barry. An a-prolog decision support system for the space shuttle. In I. V. Ramakrishnan, editor, *Proc. of the 3rd Int. Symposium on Practical Aspects of Declarative Languages (PADL'01)*, pages 169–183. Springer, 2001.
- Johannes Oetsch, Jörg Pührer, and Hans Tompits. Stepwise debugging of answer-set programs. *TPLP*, 18(1):30–80, 2018.
- Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- Kostyantyn M. Shchekotykhin. Interactive query-based debugging of ASP programs. In Blai Bonet and Sven Koenig, editors, *Proc. of the 29th AAAI Conf. on Artificial Intelligence (AAAI'15)*, pages 1597–1603. AAAI Press, 2015.
- Timo Soinen and Ilkka Niemelä. Developing a declarative rule language for applications in product configuration. In Gopal Gupta, editor, *Proc. of the First Int. Workshop on Practical Aspects of Declarative Languages (PADL'99)*, pages 305–319. Springer, 1999.
- Timo Soinen, Ilkka Niemelä, Juha Tiihonen, and Reijo Sulonen. Configuration knowledge with weight constraint rules. In Alessandro Provetti and Tran Cao Son, editors, *Proc. of the 1st Int. Workshop on Answer Set Programming (ASP'01)*, volume 1, 2001.
- Tran Cao Son, Orkunt Sabuncu, Christian Schulz-Hanke, Torsten Schaub, and William Yeoh. Solving goal recognition design using asp. In *Proc. of the 30th AAAI Conf. on Artificial Intelligence (AAAI'16)*, pages 3181–3187, 2016.

- Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time. In Alfred V. Aho, Allan Borodin, Robert L. Constable, Robert W. Floyd, Michael A. Harrison, Richard M. Karp, and H. Raymond Strong, editors, *Proc. of the 5th Annual ACM Symposium on Theory of Computing (STOC'73)*, pages 1–9. ACM, 1973.
- Larry J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):1–22, 1976.
- Juha Tiihonen, Timo Soininen, Ilkka Niemelä, and Reijo Sulonen. A practical tool for mass-customising configurable products. In *Proc. of the 14th Int. Conf. on Engineering Design (ICED'03)*, 2003.
- Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.
- Daniel Tunkelang. Faceted search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1), 2009.
- Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8(2):189–201, 1979.
- Marina De Vos, Doga Gizem Kisa, Johannes Oetsch, Jörg Pührer, and Hans Tompits. Annotating answer-set programs in lana. *TPLP*, 12(4-5):619–637, 2012.
- Celia Wrathall. Complete sets and the polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):23–33, 1976.