Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Main Track

On Abstracting over the Irrelevant in Answer Set Programming

Zeynep G. Saribatur[1] , **Matthias Knorr**[2] , **Ricardo Gonçalves**[2] , **João Leite**[2]
[1]Institute of Logic and Computation, TU Wien
[2]NOVA LINCS, NOVA University Lisbon
zeynep.saribatur@tuwien.ac.at {mkn, rjrg, jleite}@fct.unl.pt

Abstract

Generalization is an important ability that allows humans to tackle complex problems by identifying common problem structures and omitting irrelevant details. Whereas such ability comes naturally to humans, it has proved challenging to establish within AI systems. Although different research communities have tackled this challenge, their focus has usually been set on developing efficient algorithms for concrete problems, and a general theoretical understanding of the generalization ability is still lacking. In the context of Answer Set Programming (ASP), a well-established knowledge representation and reasoning paradigm for solving highly combinatorial search problems, research on generalization has primarily focused on forgetting and projection, two related operations that aim at the omission of irrelevant details, while abstraction, an operation that aims at providing a higher-level view on the common solution and problem structures, has largely been overlooked. In this paper, we develop the theoretical foundation for generalized reasoning through abstraction in ASP, focusing on the notion of abstraction through vocabulary clustering. We formally characterize when abstraction is possible, semantically define the desired result, investigate syntactic operators to obtain such abstractions, and study the computational complexity of this problem.

1 Introduction

Abstracting over and forgetting about (ir)relevant details are abilities that humans unwittingly use when tackling complex problems, enhancing their decision-making through flexibility under changing conditions, especially by generalizing to newly encountered information. While humans naturally handle such capabilities, developing them within AI systems has proven a challenging problem. For AI systems rooted in logic and Knowledge Representation and Reasoning (KRR), such capabilities are particularly important. While these systems are transparent and understandable by humans, in the sense that drawn conclusions can be concisely traced and explained, the high computational effort of reasoning imposes limits on their usage in the face of large knowledge bases. Thus, they need to acquire abstraction abilities that allow them to simplify their complex decision-making, focusing on the key elements, and to conduct generalized reasoning.

A number of different ideas have been used to approach this problem in KRR. Forgetting (Lin and Reiter 1994) focuses on finding a representation over a reduced language, thereby omitting information no longer deemed relevant, and has been studied together with closely related notions, including uniform interpolation (Visser 1996), variable elimination (Lang, Liberatore, and Marquis 2003) or ignorance (Baral and Zhang 2005), being applied for example to Description Logics (Ghilardi, Lutz, and Wolter 2006; Wang et al. 2010; Lutz and Wolter 2011), Answer Set Programming (Gonçalves, Knorr, and Leite 2023; Gonçalves et al. 2020; Eiter and Kern-Isberner 2018), Planning (Erdem and Ferraris 2007), and Modal Logic (Zhang and Zhou 2009). Another line of research focuses on induction, such as discovering higher-order abstractions for learning logic programs (Hocquette, Dumančić, and Cropper 2023), or using anti-unification for generalization computation used for inductive inference, covering First-and Higher-Order Logic, as well as Description Logics, the latter commonly under the term of least common subsumers (see (Cerna and Kutsia 2023) and references therein). Another line of research is generalized planning, where the aim is to obtain a generalized plan that can solve multiple instances of a planning problem. This was tackled, e.g., by means of abstraction (Srivastava, Immerman, and Zilberstein 2011; Illanes and McIlraith 2019) and by learning such plans (Bonet, Francès, and Geffner 2019; Bonet et al. 2019). The main focus here has been to propose algorithms that can efficiently compute such plans, without establishing a theory for generalized reasoning that could apply to other domains.

In this paper, we are focusing on building the theoretical foundations for generalized reasoning in the context of answer set programming (ASP) (Gelfond and Lifschitz 1991; Brewka, Eiter, and Truszczyński 2011), a well-established KRR paradigm for solving highly combinatorial search problems, permitting the combination of a declarative representation of these together with varying instances. Here, we are investigating generalization using abstraction by clustering the vocabulary, providing a higher-level view on the common solution and problem structures. More concretely, we are interested in finding abstractions that preserve the dependencies in a program under potential addition of facts, which resembles the notion of *uniform equivalence* (Maher 1986; Sagiv 1987; Eiter and Fink 2003) and is well-aligned with the concept of varying instances (of facts) in ASP. We illustrate our motivation in the following example.

Example 1. *Let us consider the following program P.*

$$reachHanoi \leftarrow takePlane.$$
$$reachHanoi \leftarrow takeTrain.$$
$$attendKR \leftarrow reachHanoi.$$

Since both, taking the plane and taking the train, allow us to reach Hanoi and thus attend KR, the details of which transportation is taken could be abstracted over by applying a mapping m to P that maps both $takePlane$ and $takeTrain$ to $useTransportation$ (and all other atoms to themselves) yielding the program Q below.

$$reachHanoi \leftarrow useTransportation.$$
$$attendKR \leftarrow reachHanoi.$$

We can see that Q preserves all the dependencies between the abstracted atoms and the rest of the atoms. In other words, if P is extended with further facts F over the language $\mathcal{U}$, then Q together with the abstracted facts $m(F)$ would still yield corresponding abstracted answer sets.

We note that related work on generalization in ASP, which has primarily focused on forgetting and projection, does not match this idea. Both notions have been studied in particular under uniform equivalence, the former, as UP-forgetting (Gonçalves et al. 2019; Gonçalves et al. 2021) removing certain information by restricting the admitted language of the program, and the latter under simplification via projection (Saribatur and Woltran 2023), without such restriction, but eliminating simplifiable information in the end. In addition, domain abstractions allow one to compute approximated solutions of problems that can be refined subsequently (Saribatur, Eiter, and Schüller 2021), which conceptually does not match either. Here, we focus on reducing the vocabulary through an abstraction mapping that clusters the atoms, which sets the ground for further research in the topic of generalization. Our main contributions are as follows.

- We provide the notion of uniform m-abstractions for an abstraction mapping m that clusters atoms in the vocabulary, satisfying relations resembling uniform equivalence.
- We provide necessary and sufficient conditions for testing whether a program can have uniform m-abstractions.
- We introduce a model-based representation to capture such uniform m-abstractions.
- We define a syntactic clustering operator that can obtain the abstraction whenever possible while preserving the original rules as much as possible.
- We provide complexity results for the problems of deciding uniform m-abstractability and equivalence testing.

Our paper is organized as follows. We recall preliminaries (Sect. 2), introduce, in Sect. 3, our main notion of uniform m-abstractions, with the necessary conditions for it, and provide, in Sect. 4, the semantic characterization and define the result semantically. Sect. 5 investigates a syntactic clustering method, we discuss related formalisms in Sect. 6, while complexity results are provided in Sect. 7. We close with a discussion of the potential of the notion for generalized reasoning in ASP in Sect. 8, and conclude in Sect. 9.

2 Background

We use the traditional representation of a rule r of the form

$$A_1 \vee \cdots \vee A_l \leftarrow A_{l+1}, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n,$$

where $0 \leq l \leq m \leq n$, all $A_i, 1 \leq i \leq n$, are atoms from a first-order language, and not is default negation. We also write r as $H(r) \leftarrow B(r)$ or $H(r) \leftarrow B^+(r), not\ B^-(r)$. We call $H(r) = \{A_1, \ldots, A_l\}$ the *head* of r, $B^+(r) = \{A_{l+1}, \ldots, A_m\}$ the *positive body* of r and $B^-(r) = \{A_{m+1}, \ldots, A_n\}$ the *negative body* of r. If $H(r) = \emptyset$, occasionally written as $\bot$, then r is a *constraint*; and if $B(r) = \emptyset$ and $l = 1$, then r is a *fact*.

A rule r is *normal* if $l \leq 1$, and *positive* if $B^-(r) = \emptyset$. A rule is *Horn*, if it is normal and positive.

A *disjunctive logic program* (DLP) is a finite set of rules. In the rest of the paper, we focus on propositional programs over a set of propositional atoms $\mathcal{U}$. Programs with variables reduce to their ground versions as usual. Unless stated otherwise, the term *program* refers to a disjunctive logic program.

An interpretation I is a set of atoms from $\mathcal{U}$, those atoms that are true in I. As usual, I is a model of a program P, denoted by $I \models P$, if it maps all rules of P to true. The *GL-reduct* of program P w.r.t. an interpretation I is given by $P^I = \{H(r) \leftarrow B^+(r) \mid r \in P, B^-(r) \cap I = \emptyset\}$, and I is an *answer set* of P if it is a minimal model of P^I. We denote the set of all answer sets by $AS(P)$. Two programs P_1, P_2 are *equivalent* if $AS(P_1) = AS(P_2)$, *strongly equivalent (SE)*, denoted by $P_1 \equiv P_2$, if $AS(P_1 \cup R) = AS(P_2 \cup R)$ for every program R over $\mathcal{U}$, and *uniformly equivalent (UE)*, denoted by $P_1 \equiv_u P_2$, if $AS(P_1 \cup R) = AS(P_2 \cup R)$ for any set of facts R over $\mathcal{U}$.

An *SE-interpretation* is a pair $\langle X, Y\rangle$ such that $X \subseteq Y \subseteq \mathcal{U}$; it is *total* if $X = Y$ and *non-total* otherwise. An SE-interpretation $\langle X, Y\rangle$ is an *SE-model* of a program P if $Y \models P$ and $X \models P^Y$, and we denote by $SE(P)$ the set of all SE-models of P.

A set Y of atoms is also an answer set of P if $\langle Y, Y\rangle \in SE(P)$ and no non-total $\langle X, Y\rangle \in SE(P)$ exists, and two programs P_1 and P_2 are strongly equivalent iff $SE(P_1) = SE(P_2)$ (Turner 2001). The characterization of uniform equivalence of programs can be captured by their *UE-models* (Eiter, Fink, and Woltran 2007) which are defined as $UE(P) = \{\langle Y, Y\rangle \in SE(P)\} \cup max_{\geq}\{\langle X, Y\rangle \in SE(P) \text{ and } X \subset Y\}$ where $max_{\geq}$ chooses the maximal elements of a set according to a given relation $\geq$, and $\langle X', Y'\rangle \geq \langle X, Y\rangle$ iff $Y' = Y$ and $X \subseteq X'$. Then $P_1 \equiv_u P_2$ iff $UE(P_1) = UE(P_2)$.

The following result for DLPs will be useful.

Proposition 1 ((Eiter et al. 2013))**.** *For each DLP P, it holds that for all $\langle X, Y\rangle, \langle Z, Z\rangle \in SE(P)$ s.t. $Y \subseteq Z$, $\langle X, Z\rangle \in SE(P)$.*

3 Uniform m-abstractions

In this section, we introduce abstractions as a way to generalize over programs. The main idea is to map atoms from the program language to atoms in a smaller language in such a way that the answer sets of the original program and the resulting abstraction correspond, independently of the facts,

i.e., the instance data, added to the program, in line with the notion of uniform equivalence.

To this end, we first formalize such mappings.

Definition 1. *Given sets of atoms $\mathcal{U}$, $\mathcal{U}'$ with $|\mathcal{U}| \geq |\mathcal{U}'|$, a* mapping *$m$ is a surjective function $m : \mathcal{U} \mapsto \mathcal{U}'$.*

We admit to conveniently apply mappings also to sets of sets of atoms, as well as sets of rules and/or facts and programs in a pointwise manner.

Hence, m may map several atoms from $\mathcal{U}$ to the same element in $\mathcal{U}'$. Such elements in $\mathcal{U}'$ are called *cluster atoms* and atoms in $\mathcal{U}$ mapped to these *clustered atoms*.

Definition 2. *Given a mapping $m : \mathcal{U} \mapsto \mathcal{U}'$, the set $\mathcal{U}'$ is composed of the* set of cluster atoms *$\mathcal{U}'_c = \{a \in \mathcal{U}' \mid |m^{-1}(a)| > 1\}$ and the* set of singleton atoms *$\mathcal{U}'_s = \mathcal{U}' \setminus \mathcal{U}'_c$. At the same time, $\mathcal{U}$ is composed of the* set of clustered atoms *$Cl = \{a \in \mathcal{U} \mid m(a) \in \mathcal{U}'_c\}$, and the* set of non-clustered atoms *$NCl = \mathcal{U} \setminus Cl$.*

We sometimes apply these notions also to subsets of $\mathcal{U}$ and $\mathcal{U}'$. For singleton atoms, without loss of generality, we commonly assume that they result from the identity mapping, i.e., $m(a) = a$ for $a \in \mathcal{U}'_s$. Note that this means that no atom in Cl can be mapped to any such singleton atom.

The following definition captures semantically the outlined idea of a uniform abstraction w.r.t. a mapping m.

Definition 3. *Given a program P (over $\mathcal{U}$) and a mapping m, Q (over $\mathcal{U}'$) is a* uniform m-abstraction of P *if, for any set F of facts over $\mathcal{U}$, we have*

$$m(AS(P \cup F)) = AS(Q \cup m(F)). \tag{1}$$

Example 2. *Recall program P from Ex. 1*[1] *and the indicated abstraction Q. Clearly, with $m : \{p, t\} \mapsto u$,*[2] *Q is a uniform m-abstraction of P.*

Naturally, for a fixed m, such a uniform m-abstraction may not exist.

Example 3. *Consider program P*

$$a \leftarrow not\ b \qquad b \leftarrow not\ a \qquad c \leftarrow a$$

and mapping $m : \{b, c\} \mapsto k$. Then $m(AS(P)) = \{\{a, k\}, \{k\}\}$. It is well-known that answer sets of disjunctive programs are subset-minimal, hence no corresponding program Q can exist. Now, we could be tempted to introduce programs with double negation, just like in Forgetting in Answer Set Programming (Gonçalves, Knorr, and Leite 2023) to overcome this issue, but it turns out that this does not resolve the issue. For $F_1 = \{b \leftarrow\}$, we obtain $AS(P \cup F_1) = \{\{b\}\}$, while for $F_2 = \{c \leftarrow\}$, we obtain $AS(P \cup F_2) = \{\{a, c\}, \{b, c\}\}$. Now, $m(F_1) = m(F_2) = \{k \leftarrow\}$, so, by Def. 3, we need to find a Q that with $\{k \leftarrow\}$ has simultaneously two different sets of answer sets.

This begs the question whether given a program P there always exists an m (besides the trivial one with $\mathcal{U} = \mathcal{U}'$) such that there is a uniform m-abstraction.

[1]We will abbreviate $takePlane$, $takeTrain$, $reachHanoi$, $attendKR$, $useTransportation$ by p, t, h, a, u respectively.

[2]We simplify notation and commonly only represent clustered atoms in mappings explicitly this way.

Example 4. *Consider the following program P:*

$$a \leftarrow not\ a, b$$

For $F_1 = \{a \leftarrow\}$, we obtain $AS(P \cup F_1) = \{\{a\}\}$, while for $F_2 = \{b \leftarrow\}$, we obtain $AS(P \cup F_2) = \{\}$. Since the only non-trivial m for P would be $m : \{a, b\} \mapsto k$, it follows that there are programs that have no non-trivial abstractions.

In the previous examples, we have determined the impossibility of certain results, essentially by taking advantage of the fact that sets of facts F, whose mapping results under m coincide, have a unique set of answer sets on the right-hand side of (1). Then, these different sets of facts have to have the same set of answer sets together with P (on the left-hand side of (1)). The next proposition captures this necessary condition on the answer sets of P.

Proposition 2. *If there is a uniform m-abstraction of P, then, for all Y and all sets of facts Z, Z' s.t. $m(Z') = m(Z)$:*

$$\text{If } Y \in AS(P \cup Z), \text{ then}$$
$$\exists Y' \text{ s.t. } m(Y') = m(Y) \text{ and } Y' \in AS(P \cup Z')$$

Proof. Suppose Q is a uniform m-abstraction of P, and consider any Y, and any sets of facts Z and Z' such that $m(Z') = m(Z)$ and $Y \in AS(P \cup Z)$. The latter, by (1) (of Def. 3), implies that $m(Y) \in AS(Q \cup m(Z))$. Since $m(Z') = m(Z)$ we have that $m(Y) \in AS(Q \cup m(Z'))$. Finally, using again (1), we can conclude that there is some $Y' \in AS(P \cup Z')$ such that $m(Y') = m(Y)$. □

Example 5. *Recall program P from Ex 1, and consider $m : \{t, a\} \mapsto k$. We have $\{t, h, a\} \in AS(P \cup \{t\})$, but neither $\{t, h, a\}$ nor $\{p, h, a\}$ in $AS(P \cup \{a\})$. Hence, t and a cannot be joined alone in an abstraction, only together with further atoms, namely h.*

Applying Prop. 2 this way to determine if an abstraction does not exist, would require to check this condition for all such possible sets of facts (and corresponding answer sets).

Instead, using this knowledge, we focus on obtaining necessary conditions for uniform m-abstractability in terms of SE-models that provide a more fine-grained interpretation. The main idea here is to also preserve the answer sets under different sets of added atoms. For SE-models, this corresponds to ensuring that different total and non-total SE-models, whose Y components collapse into the same under abstraction, are compatible. The conditions in Prop. 3 below capture different relevant situations as follows: $\Delta^m_{u_1}$ ensures that if there is a non-total SE-model $\langle X, Y\rangle$ that is abstracted into a total one, then there must exist another total model $\langle Y', Y'\rangle$ that allows the former to be an answer set if X is added to P[3]; $\Delta^m_{u_2}$ captures the case that if there is a total model $\langle Y, Y\rangle$ such that adding X to P would make it an answer set of P, then, for any set of facts X' that coincides with X under abstraction, some superset Y' of X', coinciding with Y under abstraction, is also an answer set; and $\Delta^m_{u_3}$ captures that, for any total model $\langle Y, Y\rangle$ of P, if Y contains one or more clustered atoms, then there is a total model containing all of those atoms abstracted to the same.

[3]Note that this total model cannot be larger by Prop. 1.

To that end, we introduce notation to refer to sets X where a non-total SE-model of a given Y that is equal or larger than X does not exist. Formally, $\nexists mod_{\supseteq}(X, Y)$ represents that, for each M with $X \subseteq M \subset Y$, $\langle M, Y\rangle \notin SE(P)$.

Proposition 3. *If there is a uniform m-abstraction of P, then P satisfies the following:*

$\Delta^m_{u_1}$: *For each $\langle X, Y\rangle \in SE(P)$ with $X \subset Y$ and $m(X) = m(Y)$, there exists $Y' \supseteq X$ with $m(Y') = m(Y)$, $\langle Y', Y'\rangle \in SE(P)$ and $\nexists mod_{\supseteq}(X, Y')$.*

$\Delta^m_{u_2}$: *For each X, Y, X' with $X \subset Y$, $\langle Y, Y\rangle \in SE(P)$, $\nexists mod_{\supseteq}(X, Y)$ and $m(X) = m(X')$, there exists $\langle Y', Y'\rangle \in SE(P)$ with $m(Y') = m(Y)$, $X' \subseteq Y'$ and $\nexists mod_{\supseteq}(X', Y')$.*

$\Delta^m_{u_3}$: *$\langle Y, Y\rangle \in SE(P)$ implies $\langle Y \cup C', Y \cup C'\rangle \in SE(P)$, where $C' = m^{-1}(m(Y \cap Cl))$.*

Proof. Let Q be a uniform m-abstraction of P.

$\Delta^m_{u_1}$: Consider $\langle X, Y\rangle \in SE(P)$ with $X \subset Y$ and $m(X) = m(Y)$, and assume no $Y' \supseteq X$ exists with $m(Y') = m(Y)$, $\langle Y', Y'\rangle \in SE(P)$ and $\nexists mod_{\supseteq}(X, Y')$. We have $\langle Y, Y\rangle \in SE(P)$, and thus $\langle Y, Y\rangle \in SE(P \cup Y)$, and $Y \in AS(P \cup Y)$. By our assumption, we cannot find a $Y' \in AS(P \cup X)$ such that $m(Y') = m(Y)$. By Prop. 2 and $m(X) = m(Y)$, we get that there is no uniform m-abstraction of P, which is a contradiction.

$\Delta^m_{u_2}$: Consider X, Y, X' with $X \subset Y$, $\langle Y, Y\rangle \in SE(P)$, $\nexists mod_{\supseteq}(X, Y)$, and $m(X) = m(X')$, and assume there is no $\langle Y', Y'\rangle \in SE(P)$ with $m(Y') = m(Y)$, $X' \subseteq Y'$ and $\nexists mod_{\supseteq}(X', Y')$. By $\nexists mod_{\supseteq}(X, Y)$, we have $Y \in AS(P \cup X)$. At the same time, by our assumption, there is no $Y' \in AS(P \cup X')$ such that $m(Y') = m(Y)$ and $m(X) = m(X')$ which by Prop. 2, indicates that there is no uniform m-abstraction of P, and we derive a contradiction.

$\Delta^m_{u_3}$: Consider $\langle Y, Y\rangle \in SE(P)$ and assume $\langle Y \cup C', Y \cup C'\rangle \notin SE(P)$, with $C' = m^{-1}(m(Y \cap Cl))$. Then there exists no $Y' \in AS(P \cup (Y \cup C'))$ with $m(Y') = m(Y)$. We have $\langle Y, Y\rangle \in SE(P \cup Y)$ and $Y \in AS(P \cup Y)$, and since $m(Y) = m(Y \cup C')$, by Prop. 2, we obtain a contradiction to Q being a uniform m-abstraction of P. □

Let us illustrate the conditions on the running example.

Example 6 (continued from Ex. 1 and 2)**.** *Consider the corresponding SE-models[4] to check the conditions in Prop. 3.*

$\langle \emptyset, ha\rangle$	$\langle a, ha\rangle$	$\langle ha, ha\rangle$	$\langle \emptyset, a\rangle$	$\langle a, a\rangle$
$\langle \emptyset, \emptyset\rangle$	$\langle \emptyset, tha\rangle$	$\langle a, tha\rangle$	$\langle ha, tha\rangle$	$\langle tha, tha\rangle$
	$\langle \emptyset, pha\rangle$	$\langle a, pha\rangle$	$\langle ha, pha\rangle$	$\langle pha, pha\rangle$
		$\langle \emptyset, tpha\rangle$	$\langle a, tpha\rangle$	$\langle ha, tpha\rangle$
		$\langle tha, tpha\rangle$	$\langle pha, tpha\rangle$	$\langle tpha, tpha\rangle$

For $\Delta^m_{u_1}$, let us examine $\langle X, Y\rangle = \langle pha, ptha\rangle$, since $pha \subset ptha$ and $m(pha) = m(ptha) = uha$. Then there is $Y' = pha$ with $m(Y) = m(Y')$, and there is no M with $pha \subseteq M \subset pha$. A similar argument can be made for the only other case $\langle tha, ptha\rangle$, thus the condition is satisfied.

For $\Delta^m_{u_2}$, let us consider $\langle p, pha\rangle \notin SE(P)$ and no larger non-total model $\langle M, pha\rangle$ exists. Since the condition is naturally satisfied for $X' = \{p\}$, we look at $X' = \{t\}$ since $m(t) = m(p) = u$. There is $\langle tha, tha\rangle \in SE(P)$ without a non-total model that is equal or larger than $\langle t, tha\rangle$. One can see that always such a Y' can be found for any non-total SE-interpretation $\langle X, Y\rangle \notin SE(P)$, and no larger SE-interpretation is a model. Thus the condition is satisfied.

For $\Delta^m_{u_3}$, we look at $\langle pha, pha\rangle$. Here the set of clustered atoms is $Cl = \{p, t\}$, and $Y \cap C = \{p\}$. The condition states for $C' = m^{-1}(m(p)) = m^{-1}(u) = \{p, t\}$, $\langle Y \cup C', Y \cup C'\rangle = \langle ptha, ptha\rangle \in SE(P)$ should hold, which is indeed the case. The same holds for $\langle tha, tha\rangle$, thus the condition is satisfied.

Given that each of the conditions in Prop. 3 are consequences of Prop. 2, we may wonder, whether there is one that implies the other. We see next that this is not the case. First, $\Delta^m_{u_1}$ does not imply the other two.

Example 7. *Consider the program from Ex. 3 and the mapping $m : \{a, b\} \mapsto k$. We have the following SE-models.*

$\langle ac, ac\rangle$	$\langle b, b\rangle$	$\langle b, bc\rangle$	$\langle bc, bc\rangle$
	$\langle \emptyset, abc\rangle$	$\langle b, abc\rangle$	$\langle c, abc\rangle$
	$\langle ac, abc\rangle$	$\langle bc, abc\rangle$	$\langle abc, abc\rangle$

In this case, $\Delta^m_{u_3}$ is not satisfied since $\langle b, b\rangle \in SE(P)$, but $\langle ab, ab\rangle$ is not. Also, $\Delta^m_{u_2}$ is not satisfied. Consider $\langle a, ac\rangle \notin SE(P)$. Yet, we have $\langle b, bc\rangle \in SE(P)$ and $m(a) = m(b)$ and $m(ac) = m(bc)$. At the same time, $\Delta^m_{u_1}$ is satisfied for the relevant models $\langle ac, abc\rangle$ and $\langle bc, abc\rangle$.

Also, $\Delta^m_{u_3}$ does not imply the other two either.

Example 8. *Consider program P with $a \leftarrow$ and $b \leftarrow not\ b$ with $m : \{a, b\} \mapsto k$, and $SE(P) = \{\langle a, ab\rangle, \langle ab, ab\rangle\}$. Then, $\Delta^m_{u_1}$ is not satisfied, since for the only non-total SE model $\langle a, ab\rangle$ there does not exist any such required Y'. Then, $\Delta^m_{u_2}$ is not satisfied either, since $\langle b, ab\rangle \notin SE(P)$, but $m(a) = m(b)$ and $\langle a, ab\rangle \in SE(P)$, but $\Delta^m_{u_3}$ clearly is.*

Finally, $\Delta^m_{u_2}$ is not more general than the other two either.

Example 9. *Consider program P from Ex. 4 with the only non-trivial mapping $m : \{a, b\} \mapsto k$ and $SE(P) = \{\langle \emptyset, \emptyset\rangle, \langle \emptyset, a\rangle, \langle a, a\rangle, \langle \emptyset, ba\rangle, \langle a, ba\rangle, \langle b, ba\rangle, \langle ba, ba\rangle\}$. $\Delta^m_{u_2}$ trivially holds for lack of missing non-total SE-models, while $\Delta^m_{u_3}$ holds by $\langle ba, ba\rangle \in SE(P)$. Yet, $\Delta^m_{u_1}$ does not hold because of $\langle b, ba\rangle \in SE(P)$.*

Alternatively, consider program P' with rules $a \vee d \leftarrow$, and $\leftarrow a$, $m : \{a, d\} \mapsto k$, and $SE(P) = \{\langle d, d\rangle\}$. Here, $\Delta^m_{u_1}$ and $\Delta^m_{u_2}$ are trivially satisfied, but $\Delta^m_{u_3}$ is not.

Yet, $\Delta^m_{u_2}$ is closely connected with UE-models, as, together with Prop. 1, we can obtain a result that establishes that certain non-total UE-models $\langle X, Y\rangle$ entail, for those Y' that coincide with Y under m, the existence of corresponding non-total UE-models $\langle X', Y'\rangle$ as follows. [5]

Proposition 4. *Let P satisfy $\Delta^m_{u_2}$. If there exists $\langle X, Y\rangle \in UE(P)$ with $m(X) \subset m(Y)$ and $X \cap Cl = Y \cap Cl$, then,*

[4] We follow a common convention and abbreviate sets in SE-interpretations such as {a,b} with the sequence of its elements, ab.

[5] For full versions of proofs throughout the paper see https://www.dbai.tuwien.ac.at/user/saribat/pub/kr24_supp.pdf.

for all $\langle Y', Y'\rangle \in SE(P)$ *with* $m(Y') = m(Y)$*, we have* $\langle (X \setminus Cl) \cup (Y' \cap Cl), Y'\rangle \in UE(P)$.

Proof sketch. Assume for $\langle X, Y\rangle \in UE(P)$ with $m(X) \subset m(Y)$ and $X \cap Cl = Y \cap Cl$, there exists $\langle Y', Y'\rangle \in SE(P)$ with $m(Y') = m(Y)$ s.t. $\langle (X \setminus Cl) \cup (Y' \cap Cl), Y'\rangle \notin UE(P)$. Then either (a) $\langle (X \setminus Cl) \cup (Y' \cap Cl), Y'\rangle \notin SE(P)$ or (b) there exists some $X' \supset (X \setminus Cl) \cup (Y' \cap Cl)$ s.t. $X' \subset Y'$ and $\langle X', Y'\rangle \in UE(P)$. For (a), if there is a greater non-total model, we apply step (b). If not, we take into account $\Delta^m_{u_2}$ which yields a contradiction due to the property on DLPs. For (b), as $\langle X', Y'\rangle$ satisfies the same conditions of the proposition, we apply the same argument to X'. Doing this recursively, eventually case (b) will not be applicable, and a contradiction by (a) will be obtained. □

Example 10. *Consider program* P *that satisfies* $\Delta^m_{u_2}$ *with* $\{\langle bx, abx\rangle, \langle abx, abx\rangle, \langle ay, aby\rangle, \langle aby, aby\rangle\} \subseteq SE(P)$ *and* $m : \{x, y\} \mapsto k$*. Then, by Prop. 4, we also have* $\{\langle ax, abx\rangle, \langle by, aby\rangle\} \subseteq SE(P)$.

This connection to UE-models is no mere coincidence, but deeply rooted in the notion of uniform m-abstractions.

4 Characterizing Abstractability

In this section, we provide a semantic characterization of uniform m-abstraction in terms of the UE-models of the given program. As we will see, this turns out to be the case, provided $\Delta^m_{u_1}, \Delta^m_{u_2}$, and $\Delta^m_{u_3}$ hold, i.e., they are sufficient conditions for the existence of a uniform m-abstraction.

To that end, we first capture those X for which, for all different total SE-models $\langle Y', Y'\rangle$ that are mapped to the same Y, and all X' compatible with X under m within Y', a UE-model exists that contains X'. The idea is that X is only considered if adding X itself or any of its variants under abstraction as facts to P does not admit an answer set.

Definition 4. *Let* P *be program,* $m\colon \mathcal{U} \to \mathcal{U}'$ *a mapping, and* $Y \subseteq \mathcal{U}'$ *such that* $\exists Y' \in m^{-1}(Y)$ *and* $\langle Y', Y'\rangle \in SE(P)$.

$$S^m_Y(P) = max_{\geq}\{X \mid m(X) \subset Y \text{ and } \forall Y' \in m^{-1}(Y) \text{ s.t. } \langle Y', Y'\rangle \in SE(P) \text{ and } \forall X' \subset Y' \text{ s.t. } m(X') = m(X) : \exists M \text{ s.t. } X' \subseteq M \subset Y' \text{ and } \langle M, Y'\rangle \in UE(P)\}$$

Here the relation $\geq$ for max is the usual superset relation $\supseteq$ on sets of atoms X.

Example 11 (Ex. 7 ctd)**.** *For* $Y = \{k\}$*, consider* $\langle b, b\rangle$ *without non-total models, thus* $S^m_{\{k\}}(P) = \emptyset$*. For* $Y = \{k, c\}$*, we have* Y' *as* $\{a, c\}$*,* $\{b, c\}$*, and* $\{a, b, c\}$*. Though, e.g.,* $\langle b, bc\rangle \in UE(P)$*, we get* $S^m_{\{k,c\}}(P) = \emptyset$*. If we add* $\langle a, ac\rangle$ *to* $SE(P)$*, then* $S^m_{\{k,c\}}(P) = \{\emptyset\}$*, and if we additionally have* $\langle ab, abc\rangle \in SE(P)$*, then* $S^m_{\{k,c\}}(P) = \{\{ab\}\}$.

Note that $m(X') = m(X)$ ensures the alignment under the abstraction. In particular, for cases where X includes some clustered atom(s), $S^m_Y(P)$ may contain different X that coincide under the mapping. This will not be an issue, as for characterizing uniform m-abstractions, we will consider the unique abstracted non-total model $\langle m(X), m(Y)\rangle$.

We now move on to defining m-UE-models, as the UE-models under an abstraction mapping m, which will be crucial for the characterization of the uniform m-abstractions.

Definition 5. *Let* P *be program and* m *a mapping. The set of* m-UE-models *of* P*, denoted* $UE_m(P)$*, is defined as:*

$$\{\langle m(X), m(Y)\rangle \mid X \in S^m_{m(Y)}(P), \langle Y, Y\rangle \in SE(P)\} \cup \{\langle m(Y), m(Y)\rangle \mid \langle Y, Y\rangle \in SE(P)\}$$

Example 12 (continued from Ex. 1 and 2)**.** *We obtain the following maximal non-total models by Def. 4:*

$$S^m_\emptyset(P) = \emptyset \qquad S^m_{\{a\}}(P) = \{\emptyset\}$$
$$S^m_{\{ha\}}(P) = \{\{a\}\} \qquad S^m_{\{uha\}}(P) = \{\{ha\}\}$$

I.e., for $\emptyset$ *no non-total model exists, for* $\{a\}$ *and* $\{h, a\}$ *only one corresponding* Y' *exists, and for* $\{u, h, a\}$ *several* Y' *exist, and among them,* $\{h, a\}$ *refers to the consensual maximal non-total model. Thus we have* $UE_m(P) = \{\langle uha, uha\rangle, \langle ha, uha\rangle, \langle a, a\rangle, \langle \emptyset, a\rangle, \langle ha, ha\rangle, \langle a, ha\rangle, \langle \emptyset, \emptyset\rangle\}$.

We now show that there is a correspondence between the m-UE-models of a program and the UE-models of its uniform m-abstraction (provided it exists). For now, this result is restricted to m creating only one cluster.

Proposition 5. *Let* P *be program and* m *a mapping such that* $|m(Cl)| = 1$*. If* Q *is a uniform* m*-abstraction of* P*, then it satisfies*

$$UE_m(P) = UE(Q). \tag{2}$$

Proof sketch. For the inclusion $UE_m(P) \subseteq UE(Q)$, we assume that $\langle X, Y\rangle \in UE_m(P)$, but $\langle X, Y\rangle \notin UE(Q)$. The non-trivial case is when $X \subset Y$, and we have two cases: there exists $\langle M, Y\rangle \in UE(Q)$ s.t. $X \subset M \subset Y$; or $\langle X, Y\rangle \notin SE(Q)$. In both cases, we reach a contradiction with the fact that $X^* \in S^m_Y(P)$ for $X^* \in m^{-1}(X)$.

For the reverse inclusion, assume that $\langle X, Y\rangle \in UE(Q)$ but suppose that $\langle X, Y\rangle \notin UE_m(P)$. The non-trivial case is $X \subset Y$. We consider $X^* = max_{\geq}(m^{-1}(X))$ and have two alternatives: X^* does not satisfy the conditions of $S^m_Y(P)$; or X^* is not maximal among those that satisfy $S^m_Y(P)$. In both cases this contradicts $\langle X, Y\rangle \in UE(Q)$. □

At the same time, we can use this characterization to show that, for arbitrary mappings m, the necessary conditions for a uniform m-abstraction in Prop. 3 are also sufficient.

Theorem 6. *Let* P *be program and* m *a mapping. If* P *satisfies* $\Delta^m_{u_1}, \Delta^m_{u_2}$*, and* $\Delta^m_{u_3}$*, then it has a uniform* m*-abstraction* Q *with* $UE(Q) = UE_m(P)$.

Proof sketch. Let P be program and m be a mapping such that P satisfies $\Delta^m_{u_1}, \Delta^m_{u_2}$, and $\Delta^m_{u_3}$. Consider Q with $UE(Q) = UE_m(P)$. Since it is possible to construct a DLP from a given set of SE-models (or UE-models), cf. Sct. 3.1 in (Eiter et al. 2013), Q is well-defined. For such P and Q, we assume that Q is not a uniform m-abstraction and thus (1) of Def. 3 cannot hold. We consider all cases that invalidate this definition and derive a contradiction for each one. □

We obtain that $\Delta^m_{u_1}, \Delta^m_{u_2}$ and $\Delta^m_{u_3}$ are indeed necessary and sufficient conditions for uniform m-abstractions.

Corollary 7. *Let P be program and m be a mapping. P has a uniform m-abstraction iff P satisfies $\Delta^m_{u_1}, \Delta^m_{u_2}$, and $\Delta^m_{u_3}$.*

In addition, for the case where $|m(Cl)| = 1$, Prop. 5 and Thm. 6 entail that (2) provides a semantic characterization of the uniform m-abstraction in terms of UE-models.

Of course, we want this characterization to hold for arbitrary m. For that we first show that satisfaction of $\Delta^m_{u_1}, \Delta^m_{u_2}$, and $\Delta^m_{u_3}$ is compositional in the following sense.

Lemma 8. *Let P be a program, Q a uniform m_1-abstraction of P s.t. $UE(Q) = UE_m(P)$, and $m = m_1 \circ m_2$ with $|m_2(Cl_2)| = 1$. If P satisfies $\Delta^m_{u_1}, \Delta^m_{u_2}$, $\Delta^m_{u_3}$, $\Delta^{m_1}_{u_1}, \Delta^{m_1}_{u_2}$, $\Delta^{m_1}_{u_3}$, then Q satisfies $\Delta^{m_2}_{u_1}, \Delta^{m_2}_{u_2}$, and $\Delta^{m_2}_{u_3}$.*

Proof sketch. Consider such P, Q, and $m = m_1 \circ m_2$ with $Cl = Cl_1 \cup Cl_2$. For each $\Delta^{m_2}_{u_i}$, the proof proceeds by considering that the respective conditions of $\Delta^{m_2}_{u_i}$ hold. Then, since $UE(Q) = UE_m(P)$, we can determine SE-models in $SE(P)$ that permit the application of the $\Delta^m_{u_i}$ and $\Delta^{m_1}_{u_i}$, respectively, which allows to show that also corresponding UE-models exist $SE(Q)$ that validate $\Delta^{m_2}_{u_i}$. □

Note that this works as intended, since any atom resulting from clustering by m_1 is not clustered further by m_2.

Thus, we can generalize the result from Prop. 5 as follows.

Theorem 9. *Let P be program and m a mapping. If Q is a uniform m-abstraction of P, then it satisfies*

$$UE(Q) = UE_m(P). \tag{3}$$

Proof sketch. If Q is a uniform m-abstraction of P, then, by Prop. 3, P satisfies $\Delta^m_{u_1}, \Delta^m_{u_2}$ and $\Delta^m_{u_3}$. Then, by Thm. 6, P has a uniform m-abstraction Q' with $UE(Q') = UE_m(P)$. We show by induction that the uniform m-abstraction is unique, i.e., $UE(Q) = UE(Q')$. □

Together with Thm. 6, this allows us to conclude that this semantic characterization is indeed unique.

Corollary 10. *Let P be program and m a mapping. Any uniform m-abstraction Q of P satisfies $UE(Q) = UE_m(P)$.*

Having established uniform m-abstractions and their characterization in terms of UE-models, we now look at the extreme cases of clustering. For the trivial mapping m (without any clusters), $\Delta^m_{u_1}, \Delta^m_{u_2}$, and $\Delta^m_{u_3}$ are always satisfied and $S^m_Y(P)$ amounts to the non-total UE-models of P. Thus, $UE_m(P) = UE(P)$, and the equality (2) amounts to $UE(P) = UE(Q)$, which is uniform equivalence.

Corollary 11. *Let m be a trivial mapping, i.e., $Cl = \emptyset$.*

- *Any program is uniform m-abstractable.*
- *Q is uniform equivalent to P iff Q is a uniform m-abstraction of P.*

On the other hand, as we have already seen in Ex. 4, for a mapping m that clusters all the elements, not every program might have a uniform m-abstraction, since some Δ condition can easily be violated (as shown in Ex. 9).

5 Syntactic Clustering

In this section, we investigate syntactic methods to provide the desired results of uniform m-abstraction. The main idea employed here is that of a syntactic rewriting, where the occurrences of the elements mapped to the same cluster will simply be replaced by that cluster, i.e., the mapping is applied to all atoms occurring in the program.

Definition 6. *Given a rule $r : H(r) \leftarrow B(r)$, $m(r)$ yields*

$$m(H(r)) \leftarrow m(B(r)). \tag{4}$$

Then, the mapping of P, *$m(P)$, is defined as $\bigcup_{r \in P} m(r)$.*

E.g., the presented result in Ex. 1 exactly matches $m(P)$.

Next, we first restrict our considerations to a class of programs for which we will show that the mapping of P provides a uniform m-abstraction.

Definition 7. *Let P be a program and m a mapping. P is called m-positive if for each rule $r \in P$, $B^-(r) \cap Cl = \emptyset$.*

Note that this only prevents clustered atoms to occur negated in the program, and Horn programs and positive programs naturally are m-positive for any m.

We first present a connection between the SE models of an m-positive program P and its mapping $m(P)$.

Lemma 12. *Let P be an m-positive program. It holds that $SE(m(P)) \subseteq m(SE(P))$.*

Proof sketch. Let $\langle X, Y\rangle \in SE(m(P))$. We claim that $\langle X_s \cup C', Y_s \cup C''\rangle \in SE(P)$ for some $C' = m^{-1}(X_c), C'' = m^{-1}(Y_c)$ exists. Assume it is not the case. Then there is at least one rule in P which prevents it from being an SE-model. Then, for all such C', C'' there is an $r \in P$ such that either $Y_s \cup C'' \not\models r$ or $X_s \cup C' \models B(r^{Y_s \cup C''})$ while $X_s \cup C' \not\models H(r^{Y_s \cup C''})$. By the construction of $m(P)$, we are able to derive a contradiction to $\langle X, Y\rangle \notin SE(m(P))$. □

This allows us to show for m-positive programs, that the mapping of P indeed is a uniform m-abstraction.

Theorem 13. *Let P be an m-positive program that has a uniform m-abstraction. Then $m(P)$ is a uniform m-abstraction of P.*

Proof sketch. We first show that $UE_m(P) = UE(m(P))$ holds for a single cluster, which by Prop. 5 and Thm. 6 entails that $m(P)$ is a uniform m-abstraction of P. We assume that this equality does not hold, and consider all cases that invalidate it, taking into account the rules in P that are modified by m, reaching a contradiction for each one. Then, following the same kind of argument as in Lemma 8 and Thm. 9, we can show that m can have several clusters. □

However, simply applying the mapping does not always work, namely when there are cyclic dependencies over negation in the rules involving atoms to cluster.

Example 13. *Consider program the P*

$$a \leftarrow not\ b \qquad\qquad b \leftarrow not\ a$$

with mapping $m : \{a, b\} \mapsto k$. Then $m(AS(P)) = \{\{k\}\}$. Clearly $Q = \{k \leftarrow\}$ is a uniform-m-abstraction. However, $m(P)$ would consist of $k \leftarrow not\ k$ without answer sets.

While this problem cannot be avoided in general, for certain programs, such as the one in the previous example, it is possible to employ elimination of negation from DLPs, building on the notion of here-total SE-interpretations.

Definition 8 ((Eiter et al. 2013)). *A set S of SE-interpretations is* here-total *iff $\langle X, Y\rangle \in S$ implies $\langle X, X\rangle \in S$.*

Example 14 (Ex. 13 ctd). *The SE-models of P are $\{\langle ab, ab\rangle, \langle a, ab\rangle, \langle b, ab\rangle, \langle \emptyset, ab\rangle, \langle a, a\rangle, \langle b, b\rangle\}$ which are not here-total as $\langle \emptyset, \emptyset\rangle \notin SE(P)$, but its UE-models are.*

This observation can be taken advantage of, since it has been shown that negation can be eliminated from a program with here-total UE-models.

Theorem 14 ((Eiter et al. 2013) adapted). *Let P be a DLP. Then, there exists a positive program P' such that $P \equiv_u P'$ iff $UE(P)$ is here-total.*

It was shown that such a P' can be obtained by left-shifting of the not-literals.

Example 15 (Ex. 13 ctd). *The UE-equivalent program of P after left-shifting of negation is $P' : a \vee b$. We then apply the mapping operator to obtain $m(P') : k$ which is a uniform m-abstraction of P' (and, therefore, also of P).*

Thus, for programs that are not m-positive but uniform m-abstractable, if their UE-models are here-total, we can create a UE-equivalent P' to apply the mapping operator.

There are still programs P which are neither m-positive nor have here-total UE-models, but still achieve a correct abstraction just using $m(P)$.

Example 16. *Consider the program P*

$$b \leftarrow a, not\ c \quad c \leftarrow b \quad a \leftarrow b \quad a \leftarrow c \quad b \leftarrow c$$

with SE-models $\langle \emptyset, \emptyset\rangle, \langle \emptyset, abc\rangle, \langle a, abc\rangle, \langle abc, abc\rangle$ and $m : \{b, c\} \mapsto k$ for which P is uniform m-abstractable. P is neither m-positive nor its UE-models here-total, but applying m to P yields a uniform m-abstraction:

$$k \leftarrow a, not\ k \quad k \leftarrow k \quad a \leftarrow k$$

Yet, if we extend P with $e \leftarrow not\ f$ and $f \leftarrow not\ e$ and the mapping with $\{e, f\} \mapsto l$, then the result is neither m-positive nor has here-total UE-models, but none of the presented methods would provide the desired abstraction. Thus, further refinements may be investigated in the future taking into account, e.g., the possibility of considering here-total UE-models only for some of the clustered atoms.

6 Related Formalisms

We now connect uniform m-abstractions to related work in ASP in the scope of generalization, namely, work on forgetting and simplifications, tailored here towards UE-models.

UP-forgetting Forgetting aims at eliminating a set of atoms from a knowledge base while preserving all relationships (direct and indirect) between the remaining atoms. In the context of ASP, this notion has been explored extensively, and a large variety of classes of forgetting operators has been defined, satisfying differing sets of properties (Gonçalves, Knorr, and Leite 2023). Central among them is a property called strong persistence, which essentially relies on the correspondence of the answer sets under strong equivalence between a program and its result of forgetting.

In this context, UP-forgetting (Gonçalves et al. 2019) emerged as a notion that establishes this semantic correspondence between a program and its forgetting result based on uniform equivalence. Formally, given a program P and a set $V \subseteq \mathcal{U}$, a forgetting operator f satisfies uniform persistence if: $\mathcal{AS}(\mathsf{f}(P, V) \cup R) = \mathcal{AS}(P \cup R)_{\|V}$, for all sets of facts R with $\mathcal{U}(R) \subseteq \mathcal{U} \setminus V$, where $S_{\|V}$ is the restriction of S to V. It was shown that there is a class of forgetting operators that satisfies this property, unlike the general case under strong forgetting (Gonçalves et al. 2020).

We can observe that even though the definitions of uniform persistence and uniform m-abstractions share certain characteristics (using uniform equivalence to compare a program and its modification), both are inherently different. Though both reduce the number of distinct atoms, forgetting removes them from the signature, while abstraction conjoins atoms if possible. In particular, an m-abstraction may not exist, while UP-forgetting is always possible.

Example 17. *Recall program P from Ex. 3 with mapping $m : \{b, c\} \mapsto k$. We know that no uniform m-abstraction exists, even if double negation is used, but uniform forgetting is applicable, for any set of atoms, including $\{b, c\}$. The answer sets of P are $\{\{a, c\}, \{b\}\}$. Here the forgetting result needs to ensure, that, e.g., with empty R the answer sets over a program not mentioning b and c still has to answer sets $\{\{a\}, \{\}\}$ which can be achieved by $a \leftarrow not\ not\ a$.*

Note that forgetting commonly requires the usage of double negation, but this is a matter of representation, which does not help in the case of uniform m-abstractions.

Finally, syntactic UP-forgetting has been investigated (Gonçalves et al. 2021), showing that this is possible as long as certain rules involving double negation are not present.

Uniform Simplification Notions of simplification (Saribatur and Woltran 2023) were introduced based on the idea to capture and preserve relations between programs and their simplifications, taking into account the entire language for the validation using some equivalence notion, unlike forgetting, which restricts to the remaining language. It turned out that this can be viewed as capturing as simplifiable atoms that semantically correspond to facts. In this setting, uniform simplification by omission (Saribatur and Woltran 2023) focuses on preserving the relevant UE-models.

Definition 9 ((Saribatur and Woltran 2023)). *Given $A \subseteq \mathcal{U}$ and a program P (over $\mathcal{U}$), program Q (over $\overline{A} = \mathcal{U} \setminus A$) is a* uniform A-simplification of P *if for any set F of facts over $\mathcal{U}$, we have $AS(P \cup F)_{|\overline{A}} = AS(Q \cup F_{|\overline{A}})$. P is uniform A-simplifiable if there exists such a program Q.*

It was shown that the SE-models of a uniform A-simplifiable program need to adhere to certain conditions, similar in spirit to those presented in Prop. 3, and, if satisfied, the simplification can be characterized by the relevant

UE-models just projecting away those atoms to be omitted. A corresponding syntactic operator is presented which omits those atoms from the program, and which is applicable whenever simplification is possible.

Note that the difference between abstraction and simplification is that, for the latter, we might have SE-models which do not contain any atoms to omit, while still agreeing on the projection with other SE-models, which cannot happen with abstraction. Thus in order to relate these two notions, we need to impose a restriction on the SE-models, namely, that all SE-models $\langle X, Y\rangle$ contain a clustered atom in X from each cluster in the mapping.

Definition 10. *Let P be a program and $m : \mathcal{U} \mapsto \mathcal{U}'$ a mapping. P is called $\mathcal{U}'_c$-*insistent *if, for each $\langle X, Y\rangle \in SE(P)$, $X \cap m^{-1}(k) \neq \emptyset$ hold for all $k \in \mathcal{U}'_c$.*

It is possible to observe that, for such restricted $\mathcal{U}'_c$-insistent P, the conditions $\Delta^m_{u_1}$, $\Delta^m_{u_2}$, and $\Delta^m_{u_3}$ reduce to those of uniform simplifiability for omitting those atoms in Cl (see (Saribatur and Woltran 2023)). We thus obtain:

Proposition 15. *Let m be a mapping and P a $\mathcal{U}'_c$-insistent program. If P is uniform m-abstractable, then P is uniform $\mathcal{U}'_c$-simplifiable.*

If we consider a single cluster only, then under these restrictions (on abstractions), these two notions correspond.

Proposition 16. *Let m be a mapping with $|\mathcal{U}'_c| = 1$ and P a $\mathcal{U}'_c$-insistent program. P is uniform m-abstractable iff P is uniform $\mathcal{U}'_c$-simplifiable.*

Note though that the actual individual results of applying simplification and abstraction do not correspond:

Example 18. *Consider the program P as*

$$a \vee b. \qquad d \leftarrow a, c. \qquad d \leftarrow b, c.$$

Now consider the abstraction mapping $m : \{a, b\} \mapsto k$. In all of the SE-models of P either a or b appears, thus P is $\{a, b\}$-insistent. Here P is both uniform m-abstractable and uniform A-simplifiable for $A = \{a, b\}$. For simplification, we would obtain a single rule $d \leftarrow c$, whereas for abstraction we have $k \leftarrow$ and $d \leftarrow k, c$.

Note that in the example above, by symmetry of a and b in the program structure, uniform m-abstraction is applicable. Since, in addition, both together could be viewed as facts, simplification also is. In general, the latter is not the case. Also, if possible, simplifying a single atom provides a simplified program, whereas abstraction amounts to a trivial mapping. Together with the inherent differences in the characterizations under UE-models, this shows that while seemingly building on syntactically similar notions, cf. Defs. 3 and 9, they represent entirely different ideas and results.

7 Complexity

We assume familiarity with basic concepts of complexity theory. For comprehensive details we refer to (Papadimitriou 2003; Arora and Barak 2009).

Theorem 17. *Let P be a program over $\mathcal{U}$ and $m : \mathcal{U} \to \mathcal{U}'$ be a mapping. Deciding whether P is uniform m-abstractable is Π^P_3-complete.*

Proof. We show Π^P_3 membership by analysing the three conditions of Proposition 3 separately. Π^P_3-hardness is obtained by knowing that uniform simplifiability, which is Π^P_3-hard, is a special case of uniform abstractability (Prop. 16).

(Δ_{u_1}) Membership: We give Σ^P_3 membership for the complementary problem: It suffices to guess an SE-interpretation $\langle X, Y\rangle$ and check whether $\langle X, Y\rangle \in SE(P)$ and $\forall Y' \supseteq X$ with $Y' \subset Y$, $m(Y') = m(Y)$ and $\langle Y', Y'\rangle \in SE(P)$, $\exists M$ with $X \subseteq M \subset Y'$, $\langle M, Y'\rangle \in SE(P)$. Inspecting the quantifier structure of this condition, and since SE-model checking is in P (Eiter, Fink, and Woltran 2007), yields the required membership.

(Δ_{u_2}) Membership: Again, we give Σ^P_3 membership for the complementary problem: It suffices to guess interpretations X, Y, with $X \subset Y$ and check whether $\langle Y, Y\rangle \in SE(P)$, and (a) exists M with $X \subseteq M \subset Y$, $\langle M, Y\rangle \in SE(P)$, or (b) exists X' with $m(X) = m(X')$, such that for all $\langle Y', Y'\rangle \in SE(P)$ with $m(Y') = m(Y)$ such that $X' \subseteq Y'$ there is an M' with $X' \subseteq M' \subset Y'$, $\langle M', Y'\rangle \notin SE(P)$. Since SE-model checking is in P, we derive that (a) is in NP; while (b) due to the quantifier structure yields a Σ^P_3 procedure; together with our overall guess for X, Y, it follows that the entire procedure remains in Σ^P_3.

(Δ_{u_3}) Membership: we solve the complementary problem, and guess $\langle Y, Y\rangle \in SE(P)$ so that $\langle Y \cup C', Y \cup C'\rangle \notin SE(P)$ for $C' = m^{-1}(m(Y \cap Cl))$ - this is in coNP. □

For checking whether some Q is a uniform m-abstraction of a given P, we give only an upper bound. The trivial lower bound is obtained by utilizing the observation in Cor. 11 where we have seen that for the trivial mapping m this checking amounts to uniform equivalence between P and Q. We anticipate that matching lower bounds can be obtained but leave this for future work.

Theorem 18. *Given a mapping m, a program P which is uniform m-abstractable, and a program Q, checking whether Q is a uniform m-abstraction of P is in Π^P_3 and Π^P_2-hard.*

Proof. For given P, Q, and m, we check for the equality (1) defining Q to be a uniform m-abstraction. For the complementary problem, we guess a set F of facts and an interpretation I checking its containment in $AS(P \cup F)$ and the containment of $m(I)$ in $AS(Q \cup m(F))$, but that it does not hold for both. As answer set computation for propositional DLPs is in Π^P_2, the equality check is contained in Σ^P_3. □

Let us look at the particular case of Horn programs to see whether deciding abstractability or checking for an abstraction becomes easier. For deciding uniform abstractability for a mapping m, we need to be checking for $\Delta^m_{u_1}, \Delta^m_{u_2}$ and $\Delta^m_{u_3}$. For $\Delta^m_{u_1}$, we can in fact take advantage of the following property on the SE-models of Horn programs: $\langle X, Y\rangle \in SE(P)$ iff $X \subseteq Y, \langle X, X\rangle \in SE(P)$ and $\langle Y, Y\rangle \in SE(P)$ (Wang et al. 2014). The existence of such $\langle X, X\rangle$ automatically satisfies $\Delta^m_{u_1}$ for any m. However the remaining two conditions $\Delta^m_{u_2}$ and $\Delta^m_{u_3}$ still need to be checked and the complexity does not seem to be reduced.

For checking whether some Q is a uniform m-abstraction of a Horn program P, we can take advantage of the fact that,

by Def. 7 and Thm. 13, Q is Horn as well, and that answer set computation for Horn programs is in P.

Theorem 19. *Given a mapping m, a Horn program P which is uniform m-abstractable, and a Horn program Q, checking whether Q is a uniform m-abstraction of P is in* coNP.

The same argument can be adapted for here-total programs (including positive programs) using Thm. 14, which ensures that any here-total program can be converted into a positive program, and that answer set computation of positive programs is in coNP.

Theorem 20. *Given a mapping m, a program P which is uniform m-abstractable and has here-total UE-models, and a program Q that has here-total UE-models, checking whether Q is a uniform m-abstraction of P is in Π_2^P.*

8 Discussion

In this section, we discuss the potential of the introduced notions to capture generalized reasoning in ASP.

Generalizing to new information The human ability of generalization from given information (or examples) to apply when encountering new information (or examples) similar to existing ones, is referred as analogical abstraction in psychology and a widely studied concept (Gentner and Hoyos 2017). In the context of ASP, being able to construct an abstraction of a given program which admits extensions with further detailsfor which the abstraction still holds would be a powerful tool for reasoning.

Example 19 (continued from Ex. 1 and 2)**.** *Consider extending P to P' over $\mathcal{U}' = \mathcal{U} \cup \{takeCar\}$ by adding the rule $reachHanoi \leftarrow takeCar$. The dependencies between $takeCar$ and the other atoms are same as these dependencies for $takePlane$ or $takeTrain$. Thus extending the original mapping m : $\{takePlane, takeTrain\} \mapsto useTransportation$ to m' : $\{takePlane, takeTrain, takeCar\} \mapsto useTransportation$ would be possible and the existing Q would remain the same as the uniform m'-abstraction.*

This observation leads to the following conjecture which for simplicity only considers one cluster and where a/e represents the replacement of a with e.

Conjecture 21. *Let P be a program over $\mathcal{U}$ and uniform m-abstractable. Consider $\mathcal{U} \cup \{e\}$. If P' is of form $P \cup \{r_{|a/e} \mid r \in P, a \in r\}$ for some $a \in Cl$, then P' is uniform abstractable for m' where $Cl_{m'} = Cl_m \cup \{e\}$.*

Observe that this syntactic check with existing rules containing clustered elements is needed to ensure that we preserve the dependencies within the propositional program. We leave investigating this for future work.

Generalized planning Generalized planning is the problem of finding a plan that can work for a set of planning problem instances. This problem is widely studied in AI, and approached under different perspectives, including abstraction. We consider this problem in the context of ASP. Planning is an important application domain for ASP, also referred as answer set planning (Tran et al. 2022). Previous work towards generalized planning aims at finding conformant plans under incomplete information (Romero, Schaub, and Son 2017). The notion of quantified ASP (Fandinno et al. 2021) also seems to have potential to capture generalized planning, which so far has not been studied. Here we approach generalized planning in ASP via abstraction.

Example 20. *Consider the blocksworld problem with multiple tables. We have 3 blocks where b_1 is located on top of table t_1, b_2 is located on top of b_3 and b_3 can be on any of the tables $t_2 - t_{10}$. Thus we have different possible initial states $I_1, \ldots, I_9$ depending on where b_3 is located. The aim is to find a plan to reach the goal state G where the blocks are piled up on table t_1. Now consider an abstraction mapping m that distinguishes the chosen table t_1 and clusters the remaining tables into $\hat{t}_2$. An abstract plan to achieve the abstract goal state $m(G)$ from the abstract initial state is $move(b_2, \hat{t}_2, 0)$, $move(b_3, t_1, 1), move(b_2, b_3, 2), move(b_1, b_2, 3)$.*

When we map this abstract plan back to any instance of the original domain, we have different possible original plans to execute (depending on which table b_2 is moved to at step 0). Any of these plans can achieve the goal from any of the initial states.

Our introduced notion can be extended to capture the above example. For this, we first would need to restrict the notion of uniform m-abstractions to a set of set of facts, describing the planning instances, and then allow only for the abstraction of these atoms of importance. This would then allow us to characterize such a generalized planning via abstraction, or along these lines generalized reasoning.

9 Conclusion

We have introduced a novel equivalence notion employing clustering-based abstraction to capture the irrelevance of details in ASP programs, while ensuring that the semantics of the original program is preserved under sets of added atoms w.r.t. the modified signature in the spirit of uniform equivalence. We have provided the necessary and sufficient conditions for such uniform m-abstractions for an abstraction mapping m in terms of the SE-models, and a semantic model-based characterization for the uniform m-abstractions. We have also investigated syntactic clustering that, for certain program classes, can essentially be achieved by merely applying the mapping to the atoms of the program, and we have studied the computational complexity of deciding uniform m-abstractability and equivalence testing.

We have also discussed the potential of the notion in terms of capturing generalized reasoning in ASP, which is a line of research we plan to continue. As the current notion takes into account clustering of atoms that might not be intuitive, e.g.. clustering $takeTrain$ and $reachHanoi$ in Ex. 1 is admissible, we plan to extend our work to allow for only certain types of atoms to be clustered. For that, a relativized version of our notion restricting the language of the further atoms to be added may be helpful, similar to the relativization of the simplification notion which allowed to capture

forgetting (Saribatur and Woltran 2024), and then employ this in the context of modularity in ASP (Oikarinen and Janhunen 2008). Such extensions would also allow us to tackle another important issue we left for future work, namely how to find the right abstractions automatically, which, without any guidance, is likely unfeasible. Here, complementarily, also inductive logic programming for ASP, taking advantage of tools such as ILASP (Law, Russo, and Broda 2018), may prove useful to find abstractions for families of instances.

Other avenues of future work include the idea of investigating a notion that may permit to tackle clustering (of some atoms) and forgetting (of other atoms) simultaneously, or lifting these results to the non-ground case, possibly taking into account observations from domain clustering (Saribatur, Eiter, and Schüller 2021), which then could also provide a path towards generalized planning and its theoretical foundations.

Acknowledgements

We thank the anonymous reviewers for their valuable feedback. Z. G. Saribatur has been supported by the Austrian Science Fund (FWF) project T-1315. R. Gonçalves, M. Knorr, and J. Leite have been supported by project NOVA LINCS ref. UIDB/04516/2020 (https://doi.org/10.54499/UIDB/04516/2020) as well as by project NOVA LINCS ref. UIDP/04516/2020 (https://doi.org/10.54499/UIDP/04516/2020) with the financial support of FCT.

References

Arora, S., and Barak, B. 2009. *Computational complexity: a modern approach*. Cambridge University Press.

Baral, C., and Zhang, Y. 2005. Knowledge updates: Semantics and complexity issues. *Artificial Intelligence* 164(1-2):209–243.

Bonet, B.; Fuentetaja, R.; E-Martín, Y.; and Borrajo, D. 2019. Guarantees for sound abstractions for generalized planning. In Kraus, S., ed., *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 1566–1573. ijcai.org.

Bonet, B.; Francès, G.; and Geffner, H. 2019. Learning features and abstract actions for computing generalized plans. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 2703–2710. AAAI Press.

Brewka, G.; Eiter, T.; and Truszczyński, M. 2011. Answer set programming at a glance. *Communications of the ACM* 54(12):92–103.

Cerna, D. M., and Kutsia, T. 2023. Anti-unification and generalization: A survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, 6563–6573. ijcai.org.

Eiter, T., and Fink, M. 2003. Uniform equivalence of logic programs under the stable model semantics. In *International Conference on Logic Programming*, 224–238. Springer.

Eiter, T., and Kern-Isberner, G. 2018. A brief survey on forgetting from a knowledge representation and reasoning perspective. *KI – Künstliche Intelligenz*. Online http://link.springer.com/article/10.1007/s13218-018-0564-6.

Eiter, T.; Fink, M.; Pührer, J.; Tompits, H.; and Woltran, S. 2013. Model-based recasting in answer-set programming. *Journal of Applied Non-Classical Logics* 23(1-2):75–104.

Eiter, T.; Fink, M.; and Woltran, S. 2007. Semantical characterizations and complexity of equivalences in answer set programming. *ACM Transactions on Computational Logic (TOCL)* 8(3):17.

Erdem, E., and Ferraris, P. 2007. Forgetting actions in domain descriptions. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, 409–414. AAAI Press.

Fandinno, J.; Laferrière, F.; Romero, J.; Schaub, T.; and Son, T. C. 2021. Planning with incomplete information in quantified answer set programming. *Theory and Practice of Logic Programming* 21(5):663–679.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9(3):365–385.

Gentner, D., and Hoyos, C. 2017. Analogy and abstraction. *Topics in cognitive science* 9(3):672–693.

Ghilardi, S.; Lutz, C.; and Wolter, F. 2006. Did I damage my ontology? A case for conservative extensions in Description Logics. In Doherty, P.; Mylopoulos, J.; and Welty, C. A., eds., *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning*, 187–197. AAAI Press.

Gonçalves, R.; Janhunen, T.; Knorr, M.; Leite, J.; and Woltran, S. 2019. Forgetting in modular answer set programming. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, volume 33, 2843–2850.

Gonçalves, R.; Knorr, M.; Leite, J.; and Woltran, S. 2020. On the limits of forgetting in Answer Set Programming. *Artificial Intelligence* 286:103–307.

Gonçalves, R.; Janhunen, T.; Knorr, M.; and Leite, J. 2021. On syntactic forgetting under uniform equivalence. In Faber, W.; Friedrich, G.; Gebser, M.; and Morak, M., eds., *Logics in Artificial Intelligence - 17th European Conference, JELIA 2021, Virtual Event, May 17-20, 2021, Proceedings*, volume 12678 of *Lecture Notes in Computer Science*, 297–312. Springer.

Gonçalves, R.; Knorr, M.; and Leite, J. 2023. Forgetting in answer set programming - A survey. *Theory and Practice of Logic Programming* 23(1):111–156.

Hocquette, C.; Dumančić, S.; and Cropper, A. 2023. Learning Logic Programs by Discovering Higher-Order Abstractions. *arXiv preprint arXiv:2308.08334*.

Illanes, L., and McIlraith, S. A. 2019. Generalized planning via abstraction: Arbitrary numbers of objects. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7610–7618.

Lang, J.; Liberatore, P.; and Marquis, P. 2003. Propositional independence: Formula-variable independence and forgetting. *J. Artif. Intell. Res.* 18:391–443.

Law, M.; Russo, A.; and Broda, K. 2018. The complexity and generality of learning answer set programs. *Artif. Intell.* 259:110–146.

Lin, F., and Reiter, R. 1994. Forget it! In *AAAI Fall Symposium on Relevance*, 154–159. AAAI Press.

Lutz, C., and Wolter, F. 2011. Foundations for uniform interpolation and forgetting in expressive description logics. In Walsh, T., ed., *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 989–995. IJCAI/AAAI.

Maher, M. J. 1986. Equivalences of logic programs. In Shapiro, E., ed., *Third International Conference on Logic Programming*, 410–424. Berlin, Heidelberg: Springer Berlin Heidelberg.

Oikarinen, E., and Janhunen, T. 2008. Achieving compositionality of the stable model semantics for smodels programs. *Theory and Practice of Logic Programming* 8(5-6):717–761.

Papadimitriou, C. H. 2003. *Computational complexity*. John Wiley and Sons Ltd.

Romero, J.; Schaub, T.; and Son, T. C. 2017. Generalized answer set planning with incomplete information. In *Proceedings of the 10th Workshop on Answer Set Programming and Other Computing Paradigms, ASPOCP@LPNMR 2017, Espoo, Finland, July 3, 2017*.

Sagiv, Y. 1987. Optimizing datalog programs. In *Proceedings of the 6th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '87, 349–362. New York, NY, USA: ACM.

Saribatur, Z. G., and Woltran, S. 2023. Foundations for projecting away the irrelevant in asp programs. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning*, 614–624. IJCAI Organization.

Saribatur, Z. G., and Woltran, S. 2024. A unified view on forgetting and strong equivalence notions in answer set programming. In Wooldridge, M. J.; Dy, J. G.; and Natarajan, S., eds., *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, February 20-27, 2024, Vancouver, Canada*, 10687–10695. AAAI Press.

Saribatur, Z. G.; Eiter, T.; and Schüller, P. 2021. Abstraction for non-ground answer set programs. *Artificial Intelligence* 300:103563.

Srivastava, S.; Immerman, N.; and Zilberstein, S. 2011. A new representation and associated algorithms for generalized planning. *Artificial Intelligence* 175(2):615–647.

Tran, S. C.; Pontelli, E.; Balduccini, M.; and Schaub, T. 2022. Answer set planning: A survey. *Theory and Practice of Logic Programming* 1–73.

Turner, H. 2001. Strong equivalence for logic programs and default theories (made easy). In *Logic Programming and Nonmotonic Reasoning: 6th International Conference, LPNMR 2001 Vienna, Austria, September 17–19, 2001 Proceedings 6*, 81–92. Springer.

Visser, A. 1996. Uniform interpolation and layered bisimulation. In *Gödel'96*, volume 6 of *Lecture Notes in Logic*, 139–164. Springer Verlag.

Wang, Z.; Wang, K.; Topor, R. W.; and Pan, J. Z. 2010. Forgetting for knowledge bases in DL-lite. *Annals of Mathematics and Artificial Intelligence* 58(1-2):117–151.

Wang, Y.; Zhang, Y.; Zhou, Y.; and Zhang, M. 2014. Knowledge forgetting in answer set programming. *Journal of Artificial Intelligence Research* 50:31–70.

Zhang, Y., and Zhou, Y. 2009. Knowledge forgetting: Properties and applications. *Artificial Intelligence* 173(16-17):1525–1537.